FEDERAL UNIVERSITY LOKOJA

DEPARTMENT OF COMPUTER SCIENCE

B.Sc. COMPUTER SCIENCE

STUDENT HANDBOOK

2021/2022 – 2025/2026

Table of Content

Title Page

Preface

The world as we come to see it today is shaped by technologies largely made possible by computers. Looking at the advancement in technology today, one would not have believed that the first set of computers were developed less than a century ago and seem to have taken over every aspect of human endeavour.

The Computer Science Department at the Federal University Lokoja seeks to rigorously train students interested in theoretical computing and its applications to solving human problems. The curriculum is specially designed to allow students to pursue the area(s) of Computer Science that they find most interesting while providing an overall solid foundation for advanced research.

This is the third edition of the handbook for the Department. The revisions are necessary to accommodate the new changes as provided in the Minimum Academic Standards of the National Universities Commission (NUC) and provide a guide to the program and expectations for the next three or four years. In addition to providing the mission, vision, objectives and expected learning outcomes of the BSc Computer Science program; the handbook covers topics like entry and admission requirements for both Unified Tertiary Matriculation Examination (UTME) and Direct Entry (DE) candidates, graduation requirements, rules and regulations guiding the registration of courses, course listing, and detailed individual course descriptions. Also covered are information about the administration of the department, staff lists, and every other information necessary for both candidates aspiring to the program and those currently enrolled in the department.

I most sincerely welcome you to the most relevant degree program in the age of "e-everything".

Prof. Francisca O. Oladipo, **PhD, FASI, FNCS, FPASRC**
Head of Department of Computer Science
francisca.oladipo@fulokoja.edu.ng

Staff List

Teaching Staff

S/N	NAME	QUALIFICATION	SPECIALIZATION	RANK
1	Prof. Sunday Eric Adewumi	B.Sc.(UI), M.Sc. (Jos), Ph.D. (ATBU), FNCS, MCPN, C.itp	Computer Science (Numerical Algorithm)	Professor
2	Prof. Francisca Oladipo	B.Sc.(NAU), M.Sc.(NAU), Ph.D.(NAU), GTCert(MIT), FASI, Citp, FNCS, FPASRC, MCPN	Software Engineering, uComputing, Engineering of PLs, Internetware, Data Transformation, Machine Learning & Data Science	Professor
3	Dr Edgar Osaghae	B.Sc.(Uniben), M.Sc.(Uniben), Ph.D.(Uniben), FASI, MNCS, MCPN	Antivirus Systems, Malware Analysis, Algorithms, Packed Programs, Executable Packer	Reader
4	Dr Taiwo Kolajo	B.Sc.(Unilorin), M.Sc.(UI), Ph.D.(CU) PGDE(BUK), MNCS, MTRCN, FTWASRAT	Data Mining, Data Science, Big Data Analytics, Natural Language Processing	Senior Lecturer
5	Dr Emeka Ogbuju	B.Sc., M.Sc. (NAU), MNCS, MCPN	Big Data Analytics, Web/Text Mining, Data Modelling, Machine Learning	Senior Lecturer
6	Dr Fredrick Duniya Basaky	B.Sc, ABU/ ECOTES, M.SC; NOUN; Ph.D; UAS, MCPN, MNCS, MAITP	Computer/Network Security IT Engineering.	Lecturer I
6	Dr Victoria I. Yemi-Peters	B.Sc. (UDUS), M.Sc.(UNN), PGDE(UDUS), Ph.D.(KSU), MNCS, MCIA	Data Mining, Artificial Intelligence, Machine Learning, Health Informatics	Lecturer I
7	Dr Terungwa-Simon Yange	B.Sc.(BSUM), M.Sc.(OAU), Ph.D.(OAU), OCA, MCPN(Ctip), MNCS, MAITP, MDSN, MIASED, MIAENG, MNIM, MPCARE	Data Science &Engineering, Information System and Software Engineering.	Lecturer I
9	Mr Ovye Abari John	B.Sc.(NSUK), M.Sc.(Unilorin)	Computer Security, Artificial Intelligence	Lecturer II
10	Mr Ahmad Muhammed Shehu	B.Sc.(IBBU, Lapai), M.Sc.(Malaysia)	Data Mining and Software Development	Lecturer II
11	Mr Haruna Abdu	B.Sc.,M.Sc. (UMYU),MNCS	Programming, Artificial Intelligence	Asst Lecturer
12	Mr Malik Rufai Adeiza	B.Sc., M.Sc. (Uni-Agric)	Data Security, Machine Learning, Data Science	Asst Lecturer
13	Mr Olalekan Ihinkalu	Dip, B.Sc.(Jos), M.Sc.	Computer Science	Asst Lecturer
14	Mrs Temitope Oluwafemi	B.Sc.(Jos), M.Sc.(UI), MNIWIIT	Computer Science	Asst Lecturer

15	Mrs Fati Oiza Ochepa	B.Sc.(ABU), M.Sc. (UTM), MNCS, MNIWIIT	Computer Science	Asst Lecturer
16	Mr Abubakar Aliyu	B.Sc., M.Sc. (UMYU)	Computer Science	Asst Lecturer
17	Miss Memunat A. Ibrahim	BSc (FUL), MSc (ANU)	Computer Science	Grad Assistant
18	Mrs Linda O Okpanachi	BSc (FUL)	Computer Science	Grad Assistant
19	Mr Musa Kunya	BSc (BUK)	Computer Science	Grad Assistant
20	Mr Jatto A. AbdulWahab	BSc (FUL)	Computer Science	Grad Assistant
21	Mr Ahad Obansa	BSc (FUL)	Computer Science	Grad Assistant

Technical Staff

S/N	NAME	QUALIFICATION	SPECIALIZATION	RANK
1	Mr Kayode Marouf Agbaje	ND, HND, BSc, MTech (in view)	Computer Science	Chief Technologist
2	Mr Umeh Paulinus	BSc, MIT	Computer Networking	Principal Hardware Specialist
2	Mr Dauda Isiaka	HND, PGD, BSc, MSc (in view)	Computer Hardware	Senior Hardware Specialist
4	Mr Babatunde Stephen Osanaiye	BEng, MIT	Networking, Security, Artificial Intelligence	Senior Technologist

Administrative Staff

S/N	NAME	QUALIFICATION	RANK
1	Mrs Hawawu Muhammed	BSc, MPA, MSc (in view)	Administrative Officer
2	Mrs Hellen Obiageli Uwaechia	BSc, MSc (in view)	
3	Mr Ojonuba Ochagana Samuel	ND, Diploma	Senior Executive Officer (Admin)

Staff Contacts

S/N	NAME	EMAIL ADDRESS
1	Prof. Sunday Eric Adewumi	sunday.adewumi@fulokoja.edu.ng
2	Prof. Francisca Oladipo	francisca.oladipo@fulokoja.edu.ng
3	Dr. Edgar O. Osaghae	edgar.osaghae@fulokoja.edu.ng
4	Emeka Ogbuju Emmanuel	emeka.ogbuju@fulokja.edu.ng
5	Dr. Fredrick Duniya Basaky	fredrick.basaky@fulokoja.edu.ng
6	Dr. Taiwo Kolajo	taiwo.kolajo@fulokoja.edu.ng
7	Dr. Victoria I. Yemi-Peters	victoria.yemi-peters@fulokoja.edu.ng
8	Olalekan Ihinkalu Ebenezer	ihinkalu.olalekan@fulokoja.edu.ng
9	Ovye Abari John	ovye.abari@fulokoja.edu.ng
10	Ahmad Muhammad Shehu	ahmad.mohammad@fulokoja.edu.ng
11	Rufai Malik Adeiza	rufai.malik@fulokoja.edu.ng
12	Haruna Abdu	haruna.abdu@fulokoja.edu.ng
13	Mr. Olalekan Ihinkalu	olalekan.ihinkalu@fulokoja.edu.ng
14	Mrs. Temitope Oluwafemi	temitope.oluwafemi@fulokoja.edu.ng
15	Mrs. Fati Oiza Ochepa	fati.ochepa@fulokoja.edu.ng
16	Mr. Abubakar Aliyu	abubakar.aliyu@fulokoja.edu.ng
17	Miss Memunat A. Ibrahim	memunat.ibrahim@fulokoja.edu.ng

18	Mrs Linda O Okpanachi	linda.okpanachi@fulokoja.edu.ng
19	Mr. Musa Kunya	musa.kunya@fulokoja.edu.ng
20	Mr. Jatto A. AbdulWahab	jatto.abdulwahab@fulokoja.edu.ng
21	Ahad Obansa	ahad.obansa@fulokoja.edu.ng
22	Mrs Hawawu Muhammed	hawawu.mohammed@fulokoja.edu.ng
23	Mrs Hellen Obiageli Uwaechia	hellen.uwaechia@fulokoja.edu.ng
24	Ojonuba Ochagana Samuel	samuel.ochagana@fulokoja.edu.ng

1.0 History of the Department

The Computer Science Department is one of the 11 that commenced activities in the 2012/2013 academic session with the admission of forty-two (42) students, nine (9) academic staff, and one (1) non-academic staff, with Professor Sunday Eric Adewumi as the pioneer Head of Department. The initial academic programme adopted the NUC Minimum Academic Benchmark but was subsequently reviewed in 2014 and 2018 to accommodate present realities and key into the grand challenges in Computer Science.

After the graduation of the first set of students in 2016, the Department was ripe for the commencement of higher degrees. Consequently, during the 2019/2020 Academic Session, the Department welcomed the first set of students into the Postgraduate Diploma (PGD), Master of Science (MSc), and Doctor of Philosophy (Ph.D.) in Computer Science. The Department has a well-equipped computer laboratory furnished for practical applications in solving real problems for both undergraduate and postgraduate students.

Presently, the Department is staffed with qualified academic staff in diverse areas of Computer Science. They contribute to undergraduate and postgraduate training, thereby giving the students all they need to become Computer Scientists ready to take on the challenges confronting the nation and provide solutions. The leadership of the department continues to leverage their teaching and research experiences, both local and international, to garner international linkages for the University and enable cooperation with foreign institutions in many areas, including research and exchange programmes.

The Department shall continue to introduce innovative teaching and research styles to produce undergraduate and graduate students who can think critically and creatively to solve local and global real-life problems. The strategies will enable the students to be competitive, have independent thinking, and engage in successful careers anywhere. All the staff of the Department is committed to providing technical leadership by applying their computing knowledge to solving significant problems across a broad range of application areas. They can derive and use techniques, skills, and tools necessary for computing and engineering practice, offer exciting solutions that benefit humanity and the natural environment.

2.0 Philosophy

The programme, by way of providing students with a broad and balanced foundation of Computer Science knowledge and skills, aims at producing Computer Science graduates that can apply knowledge and skills for solving theoretical and practical problems in Computer Science and the development of relevant ICT for national development. They are also expected to acquire knowledge and skills to undertake further studies in Computer Science and multidisciplinary areas related to Computer Science. They should be able to acquire a range of applicable information technology skills to various aspects of human endeavours. They should demonstrate general skills relating to non-subject-specific competencies, ICT capacity, communication, interpersonal and organization skills.

2.1 Vision

To be the most outstanding Computer Science department among the nine newly established Federal Universities in Nigeria in 2011 and one of Africa's top-ranking academic programme departments.

2.2 Mission

To produce graduates who can develop and deploy cutting-edge applications geared towards solving societal challenges.

3.0 Aim

To produce graduates with in-depth knowledge and appropriate skills for high theoretical, practical, and professional proficiencies in the foundations of Computer Science and its evolving disciplines.

4.0 General Objectives

The objectives of B.Sc. Computer Science programme are:

i. to create in students the awareness of and enthusiasm for computer science and its capabilities.
ii. to involve the students in an intellectually stimulating and satisfying experience of learning and studying

iii. to provide a broad and balanced foundation in computer science knowledge and practical skills.
iv. to develop in students through an education in computer science a range of applicable transferable information technology skills to all aspects of human endeavours.
v. to generate an appreciation of the importance of computer in an industrial, economic, technological, and social context.
vi. to provide students with knowledge and skills base for further studies in computer science or multidisciplinary studies involving computer science.
vii. to acquire knowledge and skills to undertake further studies in Computer Science and multidisciplinary areas related to Computer Science.

5.0 Program Expected Learning Outcome

On successful completion of the B.Sc. Computer Science program, students will have:

i. knowledge of basic science and computer science fundamentals
ii. understanding of entrepreneurship, the need for and process of innovation, as well as the need and capacity for lifelong learning
iii. in-depth technical competence in the discipline of computer science
iv. an ability to carry out problem analysis, requirements capture, problem formulation, and integrated software development for the solution of a problem
v. capacity to continue developing relevant knowledge, skills, and expertise in computer science throughout their careers
vi. an ability to communicate effectively with other computer scientists, software engineers, other professional disciplines, managers, and the community generally
vii. an ability to undertake and coordinate large computer science projects and to identify problems, their formulation, and solution
viii. an ability to function effectively as an individual, as a team member in multidisciplinary and multicultural teams, and as a leader/manager with the capacity to assist and encourage those under their direction
ix. understanding of the social, cultural, global, and business opportunities of the professional computer scientist as well as an understanding of the need for and principles of sustainability and adaptability
x. understanding of and commitment to professional and ethical responsibilities

6.0 Programme Offered and Duration

The Department of computer science offers undergraduate courses leading to the award of the Bachelor of Science (B.Sc.) Honour Degree in Computer Science. The duration for the award of the B.Sc. (Hons.) degree shall be for four (4) years (Eight Semesters) for Unified Tertiary Matriculation Examination (UTME) candidates and three (3) years (6 Semesters) for Direct Entry (DE) candidates. However, students that fail to graduate within the normal number of sessions will not be allowed to exceed a total of six (6) years (12 Semesters), if admitted through the UTME and five (5) years (10 Semesters) if admitted through DE.

6.1 Semester Duration

A minimum of 15 weeks shall normally be reserved for teaching during each semester, excluding public holidays and semester breaks. One (1) to three (3) weeks shall be reserved for examinations after the teaching period.

7.0 Admission and Graduation Requirements

In addition to the general requirements for admission into the University, candidates intending to study **B.Sc. Computer Science** must fulfill any of the conditions below:

7.1 4-years Full-Time Degree Programme

a) The entry requirements for 100 level shall be at least five SSCE (NECO and WAEC or its equivalence) passes at credit level to include English Language, Mathematics, Physics and any other two science subjects chosen from Chemistry, Biology/Agricultural Science in not more than two (2) sittings.

b) **UTME Subjects:** The acceptable UTME subjects for admission into 100 level are English, Mathematics, Physics, and any one of Chemistry or Agricultural Science/Biology.

7.2 3-years Full-Time Degree Programme (Direct Entry)

***In addition to (a) above, candidates satisfying any of the two conditions below may undertake the three-year direct entry degree programme*:**

c) IJMB: Two A' Level passes in Mathematics, and any of Physics, Chemistry or Biology/Zoology/Botany.

d) GCE A' Level: Two A' Level passes to include Computer Science/Information Technology and any one of Physics, Mathematics, Chemistry, and Biology.

e) OND/ND/NCE/ in Computer Science/Computer Engineering/Computer Studies with at least lower credit.

7.3 Requirements for Graduation

For a candidate to be eligible for graduation and the award of a degree of Bachelor of Science in Computer Science, the candidate must have successfully completed all prescribed courses as contained in this programme curriculum and must attain the following:

i) A pass grade in Supervised Industrial Work Experience Scheme (SIWES).
ii) A minimum of CGPA of 1.00.
iii) A minimum of 128 units core courses, with a minimum of 21 units electives.
iv) A pass grade is required in all prescribed core courses of the programme.
v) A student may take some elective courses to meet the graduation requirement.

The graduation requirement for B.Sc. (Hons) Computer Science is summarized below:

	100 Level	**200 Level**	**300 Level**	**400 Level**	**TOTAL**
Core Courses	25	30	27	31	113
Electives	6	6	3	6	21
Core Course (GST)	10	5	-	-	15
TOTAL	41	41	30	37	149

The above summary table shows that for a student to graduate, he/she needs to register a total of at least **149** credit units, of which **128** credits must be core.

8.0 Requirements for Award of Degree

For a candidate to be eligible for the award of a degree of B.Sc. in Computer Science, the candidate must have successfully completed all prescribed courses as contained in the course description. The minimum number of units for the award of the degree shall be 149 units and 119 units for a **4-year** and **3-year** degree programme, respectively. These consist of:

4-years degree programme

Compulsory courses:	113 units
GST courses:	15 units
Elective courses:	21 units
Total:	**149 units**

3-years degree programme (Direct Entry)

Compulsory courses:	88 units
GST courses:	16 units
Elective courses:	15 units
Total:	119 units

9.0 Registration of Courses

9.1 General Registration Guidelines

a) Students must be aware of the time schedule for registration and must always be in possession of proper identification.

b) Student must consult with his level coordinator before filling the course registration form.

c) The Department must approve unrestricted elective courses chosen outside those listed.

d) At the point of registration, a student is expected to pay NACOSS and FOSSA dues and settle other charges as may be required from time to time.

e) De-registration of the undergraduate project is not allowed beyond the second semester.

f) Registration problems associated with ill-health may be entertained (if supported with a medical report that the University Health Service authenticates).

g) A student is regarded as bonafide only when the necessary registration forms have duly been submitted to the Departmental Registration Officer. Therefore, students are advised to adhere to registration guidelines in their own interest strictly.

9.2 Classification of Registration Courses

*Compulsory Courses (**C**):* These are courses that must be passed and used in computing the final result irrespective of the number of attempts as long as the programme permits.

Elective Courses (E): These are courses chosen by students according to their interest and on advice or guidance of their course adviser, in addition to those they must take to complete their degree requirement. It is advisable to pass the Electives because they will be used for the computation of results.

Pre-requisite: These courses must be taken and passed before the student can register for a more advanced course. The 400 Level students who have attempted a pre-requisite course but failed it can register it along with higher level course.

9.3 Workload

A Student shall normally be allowed to register for and take a minimum of 15 units and a maximum of 24 units in any Semester. This means that no student can be credited less than 30 units or more than 48 units at the end of each academic year. The 400 Level students can register up to 30 units per semester and a total of 60 units per academic session. Note that a course with 3 units implies 3 hours of lecture and 1 hour of tutorial per week.

9.4 Deferment

For a good cause, a student who wishes to defer a semester or a whole session must put a formal application to the Vice-Chancellor through the Head of Department and the Dean of Faculty for consideration and approval by the Senate. This must be done in good time for such a request to be tendered for consideration and final approval. Deferment can be sought on the following ground:

(i) Admission related issue

(ii) Ill health

(iii) Emotional stress

(iv) Other special circumstances

10.0 Examination Guidelines

Examinations are normally held at the end of each semester. Examinations may take the form of written papers, oral examinations, practical, the submission of projects, and any combination of these or any other form approved by the Senate. The Continuous Assessment (C.A.) of course work is normally included in determining examination results.

10.1 Eligibility to write End of Semester Examination

To be eligible for admission into any examination, a student must have been registered for the course unit to be examined and must have fulfilled the University requirements concerning residence, fees, or other related matters. At least 75% attendance is required in all classes,

tutorials, laboratories, etc., to qualify for examinations.

10.2 Examination Conducts

1. A student must be at the examination venue at least thirty (30) minutes before the time of the examination. A student is admitted within thirty (30) minutes after the examination has commenced but shall not be allowed extra time. On no account shall a student be allowed to leave the venue during the first hour or the last fifteen (15) minutes of the examination. A student must hand over his/her scripts to the invigilator before leaving if he/she does not intend to come back.
2. A student who leaves the examination room shall not be admitted back unless he/she has been continually under the surveillance of an Invigilator/Assistant Invigilator.
3. A student shall come along with his/her ID card and Examination Card in each examination and display them conspicuously on his desk. Each student shall complete an Attendance List bearing his/her name and matriculation number by signing during each examination.
4. No book, printed paper or written document, or unauthorized materials shall be allowed into an examination room by any student, except as stated in the rules of the examination paper. A student must not directly or indirectly give assistance to any other student during an examination or permit any other student to copy from or otherwise use his papers. Similarly, a student must not directly accept assistance from any other student or use any other student's paper.
5. Suppose any student is suspected to have infringed on any of the above provisions or in any way to have cheated or disturbed the conduct of the examinations. In that case, a report shall be made as soon as possible from the Department to the Faculty Examination Officer and the Dean. The Dean will cause the circumstances to be investigated and reported to the Board of Examiners. The student involved shall be allowed to continue with the examination provided he does not cause any disturbance; however, the Board of Examiners may subsequently recommend to the Faculty Board and Senate whether his paper should be accepted and any other action that shall be taken on the matter.
6. A student shall write his examination number and not his name distinctly in the space provided at the top of the cover of every answer booklet or a separate sheet of paper. The use of scrap paper is strictly prohibited as all rough work must be done in the answer booklet, which must be submitted to the invigilator. Except for printed question paper, the students may not remove mutilate from the examination room or any paper or other materials supplied. At the end of the time allotted for the examination, each student shall cease writing when instructed to do so and gather his/her scripts together for collection by the invigilator.

10.3 Discipline

The examination regulation set out above binds all students, the breach of which carries serious punishments prescribed below:

(i) Expulsion from the University

The following offences shall carry the punishment of expulsion:

(a) Impersonation at examinations. This may involve an exchange of examination numbers, name/answer sheets, or intentional use of someone else's examination number.

(b) Exchange of relevant materials in examination hall which may involve the exchange of question papers containing relevant jotting and materials.

(c) Exchange of answer scripts.

(d) Introduction of foreign materials to the examination hall.

(ii) Rustication for one academic year

The following offences shall carry the punishment of rustication for one academic session:

(e) Non-submission or incomplete submission of answer scripts.

(f) Collaboration/copying from each other.

(iii) Written Warning

The following offences shall attract a written warning:

(g) Speaking/conversation during examination

(h) Writing on question papers.

11.0 Grading System

Each course is examined at the end of the Semester in which it is offered. Students' progress is assessed through continuous assessment (i.e., by way of tests, written assignments, and other appropriate methods) consisting of 40% during the Semester. Examination at the end of the Semester carries 60%. Thus, the totality of every grade in each course is based on 100% marks. The score from each course is assigned appropriate letter grade as follows:

(i) Credit Units	(ii) Percentile Scores	(iii) Letter Grades	(iv) Grade Points (GPA)	(v) Grade Point Average (GPA)	(vi) Cumulative Grade Point Average (CGPA)	(vii) Class of Degree
Vary according to contact hours assigned to each course per week per semester and according to workload carried by student	70 – 100	A	5	Derived by multiplying (i) and (iv) and dividing by Total Credit Units	4.50 – 5.00	First Class
	60 – 69	B	4		3.50 – 4.49	2nd Class Upper
	50 – 59	C	3		2.40 – 3.49	2nd Class Lower
	45 – 49	D	2		1.50 – 2.39	Third Class
	40 – 44	E	1		1.00 – 1.49	Pass Degree
	0 – 39	F	0		--	--

12.0 Clear Academic Standing, Warning, Probation, and Withdrawal

The academic standing of a student is being determined by the Cumulative Grade Point Average (CGPA). The minimum CGPA is 1.00.

12.1 Clear Academic Standing

For a student to be on Clear Academic Standing, he/she should have a CGPA of not less than 1.00.

12.2 Warning

A student is warned if his/her CGPA drops below the minimum tolerable CGPA for the first time. This warning is usually in the form of verbal advice by the Level Coordinator, and the student should be made to be fully aware of the implication of dropping below the minimum tolerable CGPA in the next semester examinations.

12.3 Academic Probation

A student will be placed on Academic Probation if he/she fails to maintain a minimum CGPA of 1.00 at the end of the session. The probationary status shall be reversed if the student maintains a CGPA of at least 1.00 in any subsequent semester after the first year. The responsibility to reverse the probationary status rests with the student. A preliminary notice of poor academic standing shall be given to a student in writing by the University.

13 Withdrawal for Academic Failure

A student shall be required to withdraw for academic failure if he/she fails to maintain a CGPA of 1.00 in two (2) consecutive Academic Sessions at the end of any session.

13.0 Computation of Results

The following terminologies and abbreviations are commonly used in the computation of results.

1. **Total Registered Credit Units (TRCU): This is the summation of the Units' load** ($\sum_{i=1}^{n} U_i$) of all courses offered during the semester. For example, a student who is taking 3 courses of 3 Units each, as presented in table 2, has TRCU
   = 3 × 3 = 9 units.
2. **Total Credit Passed (TCP): This is the sum of the product of the course units and the** grade points in each for the semester ($\sum_{i=1}^{n} U_i P_i$ - where P is the Grade Point attached to the course and U is the course unit from the table in 12.1).
3. **Grade Point Average (GPA): Total Credit Passed (TCP) divided by the Total Registered Credit Unit (TRCU)**
   TCP= (3 × 5) + (3 × 4) + (3 × 2) = 33
4. **Cumulative Credit Point Average (CCPA): This is the summation of Total Credit Passed (TCP) over all semesters from the beginning to date.**

13.1 Numerical Example and Computation

100 Level First Semester Scores

Course Code	TRCU	Score	Points	TCP
CSC101	2	76	5	2 × 5 =10
MTH111	3	65	4	3 × 4 = 12
GST107	2	56	3	2 × 3 = 6

The calculation of **GPA** is as follows:

$$\boldsymbol{GPA = \frac{\sum_{i=1}^{n} TCP}{\sum_{i=1}^{n} TRCU} = \frac{(2 \times 5) + (3 \times 4) + (2 \times 3)}{2 + 3 + 2} = \frac{28}{7} = 4.00}$$

where n is the number of courses, and it is the turning variable.

100 Level Second Semester Scores

Course Code	TRCU	Score	Points	TCP
CSC102	2	68	4	2 × 4 = 8
MTH112	3	26	0	3 × 0 = 0
STA112	3	56	3	3 × 3 = 9
PHY122	2	89	5	2 × 5 = 10

$$\boldsymbol{GPA = \frac{\sum_{i=1}^{n} TCP}{\sum_{i=1}^{n} TRCU} = \frac{(2 \times 4) + (3 \times 0) + (3 \times 3) + (2 \times 5)}{2 + 3 + 3 + 2} = \frac{27}{10} = 2.70}$$

While the **CGPA** for 100 level (both first and second semester) is calculated thus:

$$\boldsymbol{CGPA = \frac{\sum_{i=1}^{n} CCPA}{\sum_{i=1}^{n} TRCU} = \frac{TCP(100L\ FIRST\ SEM) + TCP(100L\ SEC\ SEM)}{TCRU(100L\ FIRST\ SEM) + TCRU(100L\ SEC\ SEM)} = \frac{28 + 27}{7 + 10} = \frac{55}{17} = 3.24}$$

**Note: $\boldsymbol{CGPA \neq \frac{GPA(100L\ FIRST\ SEM) + GPA(100L\ SEC\ SEM)}{2} = \frac{4.00 + 2.70}{2} = \frac{6.70}{2} \neq 3.24}$

14.0 Departmental Issue Resolution Process

A student is expected to channel issues that affect him/her through their Course Level Coordinator or Academic Student Adviser. If the issue at stake is beyond the Coordinator or Student Adviser to handle, it shall be forwarded to the Examination Officer if it is academic related or to the Head of Department as the case may be. When it is impossible for any of the above to resolve the issue, the case will be reported to the Dean of the Faculty (Fig 1).

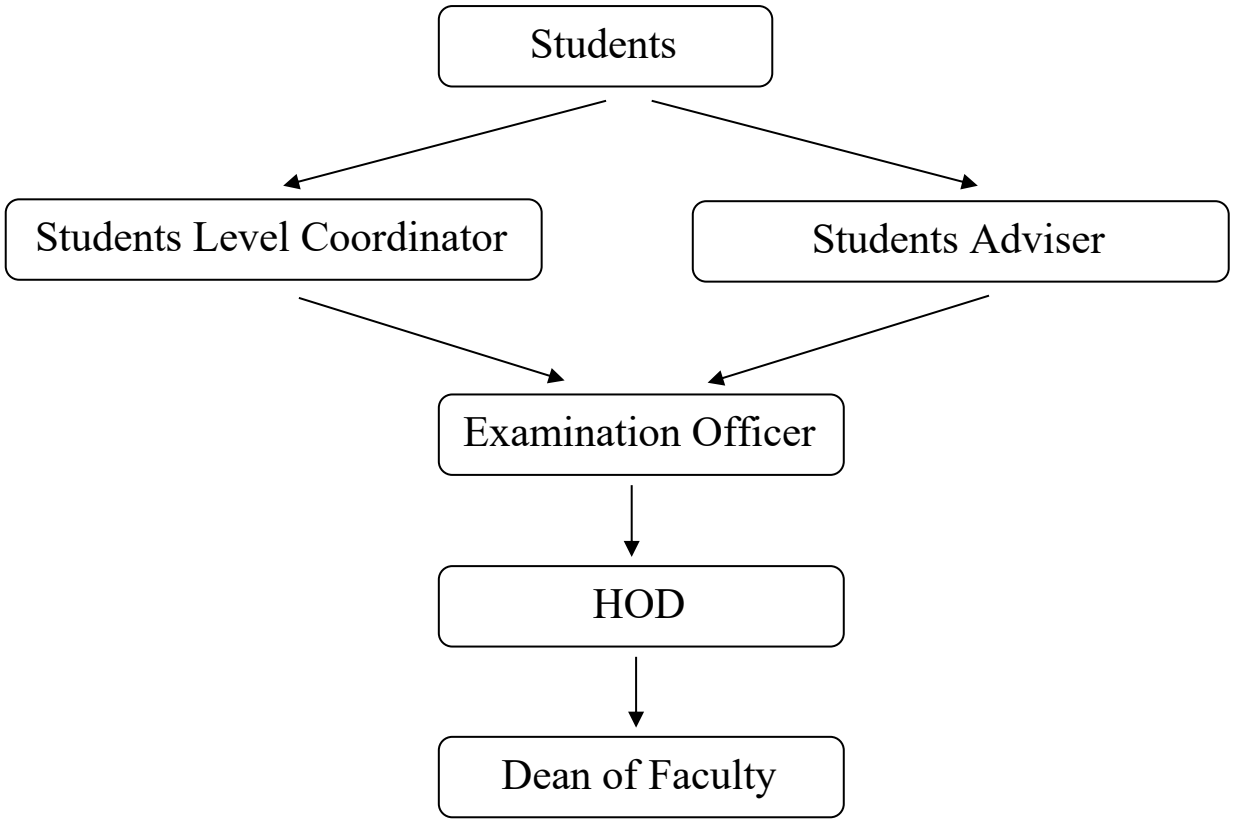

Fig 1. Hierarchy of Issue Resolution Process in the Department

15.0 Departmental Designations

Students are expected to meet their Course Level Coordinators and Departmental Student Adviser for any counseling or academic guidance. They are:

Head of Department	Prof. Francisca O. Oladipo
Examination Officer	Mr. Olalekan Ihinkanlu
Timetable Officer (Dept)	Mr. Rufai Malik Adeiza
Coordinator (100 Level)	Mrs. Temitope Oluwafemi
Coordinator (200 Level)	Mr. Abubakar Aliyu
Coordinator (300 Level)	Mr. Rufai Malik Adeiza
Coordinator (400 Level)	Dr. Taiwo Kolajo
Welfare Officer	Mr. Rufai Malik Adeiza
Research Coordinator	Dr. Emeka Ogbuju
Library Representative	Mr. Mohammed Shehu
Staff Advisers (NACOSS)	Dr. Emeka Ogbuju
Departmental Purser	Mr. Sam Ojonuba

16.0 Course Evaluation

16.1 Techniques of Students Assessment

Students should be examined by a combination of the following methods:

a) Un-announced Quizzes
b) Class Examinations
c) Home-Work Assignments
d) Mid-Semester and Final Semester Examinations.

The weights to be attached to these examinations should be determined by the Department provided that the final semester examination carried not less than 60% of the total weighting. The above methods can be carried out through any of the established techniques such as:

a) Term Papers
b) Oral presentation at examinations
c) Seminars
d) Projects
e) Written essay or objective examinations, etc.

17.0 Illness

While on campus, a student who falls sick should seek for immediate medical attention at the University Health Service. When necessary, the University Health Service may refer serious case elsewhere for further treatment. Whenever the medical condition of a student necessitates absence from academic activities, the Head of Department should accordingly be notified in writing and upon resumption for normal academic work, appropriate medical report must be presented. Any student who falls ill during an examination should immediately seek medical attention at the University Health Service and must obtain appropriate medical report and forward it to the Department (HOD) as soon as possible. If the sick student must seek for further medical assistance outside the University Health Service, the Department (HOD) must be formally informed in writing before leaving the University or Lokoja. Outside the University Campus or Lokoja (e.g., while at home or holidays) if as a result of ill-health, a student is likely to be late for registration, the Department must be informed early enough. Upon resumption, supporting evidence(s) (e.g., medical report which must be authenticated by the University Health Services) must be presented.

18.0 Course Structure

The duration of the B.Sc. (Hons.) Computer Science programme is four years. There are two semesters of formal University Studies in each academic session. At 300 Level, a student is expected to go for a six (6) months Students Industrial Work Experience Scheme (SIWES) after completion of the first semester courses, at the end of which he/she must write, present, and defend a report on what he/she learnt in the industry. At 400 Level, each student undertakes a one-year project in any field of interest besides the usual prescribed courses. A report on the project is also to be presented and defended.

The following gives a detailed breakdown of the courses in the curriculum on a semester-by- semester basis.

100 Level First Semester Courses

Code	Title	Status	Credit Unit	Prerequisite
CSC101	Introduction to Computer Science	Core	2	O/L
MTH111	Sets and Number System	Core	2	O/L Maths

MTH113	Differential and Integral Calculus	Core	2	O/L Maths
MTH115	Trigonometry and Co-ordinate Geometry	Core	2	O/L Maths
PHY111	General Mechanics	Core	2	O/L Physics
PHY161	General Physics Practical I	Core	1	O/L Physics
GST101	Communication in English and Use of Library	Core	2	O/L English
GST103	Nigerian Peoples and Cultures	Core	2	O/L English
GST107	Philosophy, Logic and Human Existence	Core	2	O/L
	Sub-Total		**17**	

CHM113	Introductory Physical Chemistry	Elective	3	O/L Chemistry
CHM161	Experimental Chemistry I	Elective	1	O/L Chemistry
PHY131	Heat & Properties of Matter	Elective	2	O/L Physics
BOT101	Plant Biology	Elective	2	O/L Biology
STA113	Probability Theory I	Elective	2	O/L Maths

Note: *A minimum of three (3) units elective is required.*

100 Level Second Semester Courses

Code	**Title**	**Status**	**Credit Unit**	**Pre-requisite**
CSC102	Introduction to Computer Applications	Core	2	O/L
MTH112	Algebra	Core	2	O/L Maths
MTH114	Conic Sections and Applications of Calculus	Core	2	O/L Maths
MTH116	Vectors and Dynamics	Core	2	O/L Maths
STA124	Probability Theory II	Core	3	O/L Maths
PHY122	Electricity, Magnetism and Modern Physics	Core	2	O/L Physics
PHY162	General Physics Practical II	Core	1	O/L Physics
GST102	Communication in English	Core	2	O/L English
GST104	Communication in French and Arabic	Core	1	O/L
GST110	History and Philosophy of Science	Core	1	O/L
	Sub-Total		**18**	

CHM124	Introductory Inorganic Chemistry	Elective	3	O/L Chem
CHM162	Experimental Chemistry II	Elective	1	O/L Chem
BIO102	General Ecology	Elective	2	O/L Bio

Note: *A minimum of three (3) units elective is required.*

100 LEVEL COURSE SUMMARY: A MINIMUM OF 41 CREDIT UNITS

Core courses (Departmental)	:	25 units
Core courses (General Studies)	:	10 units
Electives	:	6 units
Level Total	**:**	**41 units**

200 Level First Semester Courses

Code	Title	Status	Credit Units	Prerequisite
CSC203	Discrete Structures	Core	3	MTH111
CSC205	Digital Logic Design	Core	3	CSC101
CSC211	Computer Programming I	Core	2	CSC101
MTH221	Mathematical Methods I	Core	3	MTH112
PHY211	Mechanics	Core	3	-
GST205	Environmental Health	Core	1	-
	Sub-Total		**15**	

STA211	Probability Theory III	Elective	3	STA124
MTH213	Real Analysis I	Elective	3	MTH111 or equivalent
MTH217	Linear Algebra I	Elective	3	MTH112 or equivalent
MTH219	Numerical Analysis I	Elective	3	MTH113 or equivalent
MTH221	Number Theory	Elective	3	MTH111

Note: *A minimum of three (3) units elective is required.*

200 Level Second Semester Courses

Code	Course Title	Status	Credit Units	Prerequisite
CSC204	Computer Organization and Assembly Language	Core	3	CSC101
CSC206	Human Computer Interaction	Core	2	CSC101
CSC208	Artificial Intelligence I	Core	3	CSC102
CSC212	Computer Programming II	Core	2	CSC101
CSC222	Computer Electronics	Core	3	CSC101
MTH224	Introduction to Numerical Analysis	Core	3	MTH111or 113
GST202	Peace and Conflict Resolution	Core	2	-
GST204	Entrepreneurial Skills	Core	2	-
	Sub-Total		**20**	

CSC202	Computer Programming III	Elective	3	CSC101
MTH212	Ordinary Differential Equations	Elective	3	MTH114 or equivalent
MTH218	Linear Algebra II	Elective	3	MTH112 or equivalent
STA212	Probability Distribution II	Elective	3	STA112

Note: *A minimum of three (3) units elective is required.*

.

200 LEVEL COURSE SUMMARY: A MINIMUM OF 40 CREDIT UNITS

Core courses (Departmental)	:	30 units
Core courses (General Studies)	:	5 units
Electives	:	6 units
Level Total	**:**	**41 units**

300 Level First Semester Courses

Code	Course Title	Status	Credit Units	Prerequisite
CSC301	Structured Programming	Core	3	CSC211
CSC303	Fundamentals of Data Structures	Core	3	-
CSC305	Compiler Construction I	Core	3	
CSC307	Database Management	Core	3	-
CSC311	Algorithm and Complexity Analysis	Core	3	CSC203
CSC315	Computer Architecture and Sequential Program	Core	3	CSC205
CSC321	System Analysis and Design	Core	3	-
	Sub-Total		**21**	

CSC313	Operations Research	Elective	3	-
CSC319	Information Technology Law	Elective	3	-
MTH321	Mathematical Modeling	Elective	3	MTH221
MTH329	Introduction to Operations Research	Elective	3	MTH217

Note: *A three (3) units elective is required.*

300 Level Second Semester Course

Code	Course Title	Status	Credit Units	Prerequisite
CSC398	SIWES (Industrial Training)	Core	6	-

Note: *Students going for SIWES must have earned 60 credit units (30 credit units for Direct Entry Students) at the end of 200 Level Second Semester.*

300 LEVEL COURSE SUMMARY: A MINIMUM OF 30 CREDIT UNITS

Core courses (Departmental)	:	27 units
Electives	:	3 units
Total	:	30 units

400 Level First Semester Courses

Code	Course Title	Status	Credit Units	Prerequisite
CSC401	Software Development and Engineering	Core	3	CSC321
CSC403	Survey and Organization of Programming Languages	Core	4	CSC301
CSC405	Operating Systems I	Core	3	-
CSC407	Machine Learning/Data Science	Core	3	CSC208
CSC409	Net-Centric Computing	Core	3	-
	Sub-Total		16	

MTH422	Optimization Theory	Elective	3	MTH329
CSC411	Introduction to Cryptography	Elective	2	-
CSC433	Computer Graphics and Visualization	Elective	2	-
CSC421	Information Technology Project Management	Elective	2	
CSC441	Artificial Intelligence II	Elective	2	CSC208

Note: *A minimum of two (2) units elective is required.*

400 Level Second Semester Courses

Code	Course Title	Status	Credit Units	Prerequisite
CSC402	Data Communications and Networks	Core	3	CSC206
CSC404	Operating Systems II	Core	3	-
CSC412	Compiler Construction II	Core	3	CSC305
CSC400	Research Project	Core	6	-
	Sub-Total		**15**	

MTH446	System Theory	Elective	3	-
MTH448	Mathematical Modeling	Elective	3	MTH221 or MTH224
CSC406	Special Topics in Computer Science	Elective	3	-
CSC422	Computer System Performance Evaluation	Elective	2	-
CSC424	Modeling and Simulation	Elective	3	-
CSC408	Expert Systems Technology	Elective	2	-
CSC432	Formal Methods in Software Engr	Elective	3	-

Note: *A Minimum of four (4) units elective is required.*

400 LEVEL COURSE SUMMARY: A MINIMUM OF 32 CREDIT UNITS

Core courses (Departmental)	:	31 Units
Electives	:	6 Units
Total	:	37 Units

19.0 Course Contents And Description

19.1 100 Level First Semester

CSC101 - Introduction to Computer Science (2 units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE		
Faculty	Sciences	
Department	Computer	
Course Title	Introduction to Computer Science	
Study Year	1	
Course Code	CSC101	
Credit Hours	8	
Pre-requisite		
Mode of Assessment	Lecture, Assessment and Practical	
Assignment Test Final Examination Total		20% 20% 60% **100%**
Course Lecturer and Instructor	Prof. Sunday Eric Adewumi Dr Frederick Duniya Basaky Mr Malik A. Rufai Mr Abubakar Aliyu Mr Musa Kunya	
Course Description	Introducing the students to computer, History and overview of computer generations, types of computer, the hardware and software.	
Course Objectives	At the end of the study student should be able to; - Reproduce the history and background of computer, - The evolution and generations of computer - Identify the hardware and the software of computer, - Differentiate between the hard and the software - Mention the different types of computer networks - List and draw the different network topologies	
Learning Outcome	Students became conversant with computer units, know the different types of computer, could now differentiate between soft and hardware, know historical background and technology of computer	
Detailed course contents	Definition and the history and background of computer, The evolution and generations of computer Identify the hardware and the software of computer, the front and rear views of a computer unit, different types connectors Differentiate between the hard and the software Mention the different types of computer networks, List and draw the different network topologies	
Course contents Sequencing		
Week1	Introduction of the course and course outlines, Definition and the history and background of computer	
Week 2 and 3	The evolution and generations of computer Identify the hardware and the software of computer, the front and rear views of a computer unit, different types connectors	
Week 4	The evolution and generations of computer Identify the hardware and the software of computer, the front and rear views of a computer unit, different types connectors	

19.2 100 Level Second Semester

CSC102 - Introduction to Computer Applications (2 units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Introduction to Computer Applications
Year of Study	I
Course Code	CSC102
Credit Hours	2
Contact Hours	72
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight %**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	• Prof Francisca O. Oladipo • Mr. Malik A. Rufai • Mrs. Linda O.Okpanachi • Mr Abdulwahab A. Jatto
Course Description	This course is designed to introduce students to the use of the PC for business and personal purposes. Tools to be deployed in this course includes a variety of word processing software, spreadsheet applications, Windows and UNIX-Like Operating Systems, and Graphic Tools. The course is experimental in nature, so most of the learning activities will take place in the Software laboratory.
Course Objectives	At the end of this course, students should be able to: a) Understand the computer software is all about b) State the basic types of computer software c) Describe system software and application software - Define software. - Categories of software. - Examples - Applications
Learning Outcomes	At the end of the course, students will be able to: 1. Understand Operating Systems: Fundamentals Definition, Functions, Features, and Examples. 2. Microsoft Windows Operating Environment, and Windows Accessories. Booting (Types) and Shutting Down the Computer, Using the Mouse and Keyboard. 3. Files and Folders/Directories 4. Computer System Protection (Virus, Trojans, and Worms) 5. Understand Application Software (Definition, Types, examples).

	6. Application Software (Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Microsoft Access, and Microsoft Publisher). 7. Effectively use a wide range of Office applications. 8. Decide on the appropriate office productivity software for a given situation. 9. Design and present simple documents using office productivity applications. 10. Operate a variety of advanced spreadsheet, operating system and word processing functions and solve a range of problems using office productivity applications 11. Adapt quickly to new software releases.
Teaching and Learning	The class will meet for two hours each week for theoretical classes and four hours each week for practical/laboratory classes. The contact time will be used for a combination of Lectures, Recitations, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on Application Software (Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Microsoft Access, and Microsoft Publisher).
Detailed Course Content	Brief introduction to computer system. Computer software, types of software (system and application software: definition, tyes, example and applications). Introduction to the use of the PC for business and personal purposes. Tools to be deployed in this course includes a variety of word processing software, spreadsheet applications, Windows and UNIX-Like Operating Systems, and Graphic Tools. The course is experimental in nature, so most of the learning activities will take place in the Software laboratory.

Course Content Sequencing

Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	Brief introduction to computers, computer software (System and Application) • **Definition** • Functions • Types • Features • Examples • applications	**3 Hours**
Week2	System software: Microsoft Windows Operating Environment, and Windows Accessories. Booting (Types) and Shutting Down the Computer, Using the Mouse and Keyboard	**3Hours**
Week3	Files and Folders/Directories	**12 Hours**
Week4	Computer System Protection (Virus, Trojans, and Worms) **Continuous Assessment II**	**12 Hours**
Week 5,6	Application software: Microsoft Word	**12 Hours**
Week 7,8,9	Microsoft Excel	**12 Hours**

Week 10	Microsoft PowerPoint	**6 Hours**
Week 11,12	Microsoft Access	**12 Hours**
After Week 12	Examination	
Recommended Reading Material 1. **CSC102 Experimental Laboratory Manual**		

19.3 200 Level First Semester

CSC203 - Discrete Structures (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Discrete Structures
Year of Study	II
Course Code	CSC203
Credit Hours	3
Contact Hours	33
Pre-requisite(s)	MTH111
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Prof. Snday E. Adewumi Mr. Ihinkalu Olalekan Ebenezer & Mr. Dauda Isiaka -Laboratory Instructor**
Course Description	Computer Science involves the study of how computer solve problems. Computer use discrete structures to represent and manipulates data. This implies that problems to be solved by computer are represented and resolved using the ideas of discrete structures.
Course Objectives	This course would enable the understanding of the following:

	1. Apply formal logic proofs and/or informal, but rigorous, logical reasoning to real problems, such as predicting the behavior of software, analyze the time-complexity of an algorithm or solving problems such as puzzles. 2. Demonstrate comprehension of discrete structures and their relevance within the context of computer science, in the areas of data structures and algorithms, in particular. 3. Apply discrete structures into other computing problems such as formal specification, verification, databases, artificial intelligence, and cryptography. 4. Demonstrate mathematical skills, analytical and critical thinking abilities. 5. Know how to represent and deploy different discrete structures to manipulate problems (data), and communicate technical results clearly and effectively using the technical language of the field correctly. 6. Prove computational theorems, propositional and predicate logic theorems.
Learning Outcomes	At the end of the course, students will be able to: 12. From this course student will learn the fundamental algorithms used by computer programmers. 13. Students will need a solid background in these subjects, because it is an excellent tool for improving reasoning and problem-solving skills. 14. Concepts and notations from discrete mathematics are useful in studying and describing objects and problems in all branches of computer science, such as computer algorithms, programming languages, cryptography, automated theorem proving, and software development. 15. computer implementations are tremendously significant in applying ideas from discrete mathematics to real-world applications, such as in operations research.
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures, Seminar Presentation, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving and software modelling using Python
Detailed Course Content	Review on Basic structures in discrete structure: set theory -basic definitions, operators and operations, set theory applications -relations: functions within sets, Application of relations. Boolean Algebra: -concepts, operators and operations, axioms (postulates) and theorems, applications. Graph theory: -concepts and theorems, graph representations, operations, applications, trees. Matrices: -concepts and theorems, operators and operations, lattices. Discrete probability: -concepts and theorems, operations, applications. Counting: -rules/theorems, operations, applications. Proof techniques, including the structure of mathematical proofs, direct proofs, disproving by counterexample, proof by contradiction. Basics of counting, including counting arguments, the pigeonhole principle, permutations and combinations, solving recurrence relation. Discrete probability, including finite probability space, axioms of probability, conditional probability.

Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	• Introduction of the course Discrete Structure • Set Theory	**3 Hours**
Week2,3,4	• Boolean Algebra • Graph theory • Matrices **1. Continuous Assessment I**	**9 Hours**
Week 5,6,	• Probability • Discrete probability and Conditional Probability • Counting Techniques	**6 Hours**
Week7,8	• Mathematical Induction • Permutations and Combinations	**6 hours**
Week9,10,11	• Functions and Mapping • Practical sessions with Python Programming 2. **Continuous Assessment II**	**9 Hours**
After Week 12	3. Examinations	

Recommended Reading Material

1. Keneth Rosen (2018). Discrete Mathematics and Its Applications. 8th ed. ISBN: 1260091996.
2. Susanna S. E. (2019). Discrete Mathematics with Applications. 5th ed. ISBN:1337694193.
3. White, R. T. (2021). Practical Discrete Mathematics. ISBN: 1838983147/
4. Levin, O. (2018). Discrete Mathematics: An Open Introduction. 3rd ed. ISBN: 1792601690

CSC205 – Digital Logic Design (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Digital Logic Design
Year of Study	II
Course Code	CSC205
Credit Hours	3
Contact Hours	36
Pre-requisite(s)	CSC101

Mode of Delivery	Classroom Lectures

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Mr. Ihinkalu Olalekan Ebenezer** **Mr. Jatto Abdulwahab**
Course Description	This is a core course in computer science that presents basic tools for the design of digital circuits. It serves as a building block in many disciplines that utilize data of digital nature like digital control, data communication, digital computer etc.
Course Objectives	This course would enable the understanding of the following: 1. Perform arithmetic operations in many number systems. 2. Manipulate Boolean algebraic structures. 3. Simplify the Boolean expressions using Karnaugh map and Quine-McClusky (Tabulation) method. 4. Implement the Boolean functions using gates (OR, NOT, AND, NAND, NOR, XOR etc.) 5. Analyze and design various combinational logic circuits 6. Understand the basic functions of flip flops. 7. Understand the importance of state diagram representation of sequential circuits. 8. Understand how to reduce and assign states in a sequential circuit. 9. Analyze and design clocked sequential circuits. 10. Able to understand and use one high-level hardware description languages (VHDL or Verilog) to design combinational or sequential circuits.
Learning Outcomes	At the end of the course, students will be able to: 1. An ability to apply knowledge of mathematics, science, and engineering. 2. An ability to design and conduct experiments, as well as to analyze and interpret data. 3. An ability to design a system component, or process to meet desired needs within realistic constraints. 4. An ability to identify, formulate, and solve engineering and real life problem. 5. An ability to use the techniques, skills and modern engineering tools necessary for engineering practice.
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures, Seminar Presentation, Tutorials. Key concepts would be taught during instructor-led sessions,
Detailed Course Content	Digital Systems and Binary Numbers: Digital systems, binary numbers, number-base conversion, octal and hexadecimal numbers, complements, signed binary numbers, binary codes, binary storage and registers, binary logic. Boolean Algebra and Logic Gates: Introduction, Basic definitions,

	Axiomatic definition of Boolean Algebra, Basic theorems and properties of Boolean Algebra, Boolean functions, canonical and standard forms, other logic operations, Digital logic gates, Integrated circuits. RTL&DTL Circuits, Integrated Injection Logic, Transistor-Transistor Logic, Emitter Coupled Logic, MOS &CMOS. Gate Level Minimization: The map method (Karnaugh map), Two-variable map, Three-variable map, Four-variable map, Five-variable map, product of sums simplification, Don't care conditions, NAND and NOR implementation, other Two-level implementations, Exclusive-OR function. Combinational Logic: Introduction, Combinational circuits, Analysis procedure, Design procedure, Binary adder-subtractor, Decimal Adder, Binary multiplier, Magnitude Comparator, Decoders, Encoders, Multiplexers. Combinational Logic**:** Design procedure, adders, subtractors, code conversion, analysis procedure, multilevel NAND & NOR circuits, exclusive-OR & equivalence functions. Binary parallel adder, decimal adder, magnitude comparator, decoder, multiplexer, programmable logic array. Synchronous Sequential Logic: Introduction, Sequential circuits, Storage element; Latches and Flip-flops, Analysis of clocked sequential circuits, State reduction and assignment, Design procedure. Flip flop, triggering of flip-flop, state reduction & assignment, design procedure, design of counters, design of state condition. Registers, Counters and the Memory Unit; Inter-register transfer, shift register, conditional control statements, overflow, decimal & floating point data, non-decimal data, Modulus N counters, memories, ROM, EPROM, PROM and RAM, dynamic RAM.

Course Content Sequencing

Weeks	Detailed Course Outline	Allocated Time
Week1	• History and Introduction to Computing Systems.	**3 Hours**
Week2,3,4	• Mechanical Computers, Digital versus Analog systems • Organization of stored Program Digital Computer • Information representation and Number systems **1. Continuous Assessment I**	**9 Hours**
Week5,6,7	• Boolean Algebra & Switching Theory(Fundamentals on Boolean Algebra; Basic Postulates, Diagrams and Theorems of Boolean Algebra, Boolean Operators.	**9 Hours**
Week8,9	2. **Seminar Presentation** • Switching Functions; Switching Circuits, Physical properties of Gates. • Combinatorial Circuits	**6 Hours**
Week10,11,12	• Analysis of sequential Circuits • Introduction to Sequential MSI's and Programmable Logic Devices 3. **Continuous Assessment II**	**9 Hours**
After Week 12	4. Examinations	

Recommended Reading Material

1. Introduction to Digital Logic Design: Solutions Manual by John P. Hayes
2. C. S. French "Introduction to Computer Science" Fifth Edition
3. Schaum's Outline "Boolean Algebra and Switching Circuits"

4. https://www.electronics-tutorials.ws/logic/logic_1.html

CSC211: Computer Programming I (2 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Computer Programming I
Year of Study	II
Course Code	CSC211
Credit Hours	2
Contact Hours	24
Pre-requisite(s)	CSC101
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Fati Oiza Ochepa (Mrs), Linda Ojone Okpanachi (Mrs)** **Mr. Paulinus Umeh -Laboratory Instructor**
Course Description	This course is intended for those with little or no programming background, though prior programming experience will make it easier, and those with previous experience will still learn C++ specific constructs and concepts. In this course, students will learn the basics about C++ programming language such as variables, data types, arrays, pointers, functions and classes etc.
Course Objectives	This course would enable the understanding of the following: 1. Basic programming concepts. 2. Train the students in the process of creating Visual C++ solutions that include forms with controls and code that make the forms functional Develop in the students, the abilities to apply, build and modify rule-based systems to solve real problems, 3. Equip students to be able to validate input and format output. 4. Develop an algorithm to solve a given problem. The student can translate an algorithm into a program that includes forms and Visual

	Basic code. Students can recognize and use recommended programming style and technique. 5. Write Windows applications using forms, controls, and events.
Learning Outcomes	At the end of the course, students will be able to: Upon completion of this course, the student will be able to: 16. Utilize the Microsoft Visual C++ integrated development environment to create Visual C++ solutions that include forms with controls and code that make the forms functional. The integrated debugger can be used to find syntax and run time errors; or any other available editor like C Free, Dev etc. 17. Create projects with forms that may include labels, picture boxes, textboxes, group boxes, list boxes, check boxes, and radio buttons and modify applicable control properties. Event procedures can be developed to allow a user to interact with the form. Students will be able to create applications with multiple forms and menus. 18. Effectively utilize decision structures such as If…Then…Else and Select…Case statements, loops and nested loops, logical operators, relational operators, arithmetic operators, procedures and functions. 19. Develop an algorithm to solve a given problem. The student can translate an algorithm into a program that includes forms and Visual Basic code. Students can recognize and use recommended programming style and technique. 20. Create one and two dimensional arrays for sorting, calculating, and displaying of data. 21. Write C++ programs using object-oriented programming techniques including classes, objects, methods, instance variables, composition, and inheritance, and polymorphism. 22. Write Windows applications using forms, controls, and events.
Teaching and Learning	The class will meet for two hours each week. Class time will be used for a combination of Lectures, Recitations, and Tutorials while Laboratory Practical Sessions will be held four hours weekly, as students are divided into groups because of their size. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving and software modelling using Microsoft Visual C++ integrated development environment.
Detailed Course Content	Introduction-Comparison of procedure-oriented, structure-oriented, event-driven, and object-oriented programming paradigms, Fundamental of object oriented design. Features of object-oriented programming. C++ Fundamentals-Keywords, data types, standard I/O streams, function prototypes, C++

	enhancements over C, default function parameters, inline functions, overloaded functions, reference variables comparison between pointers and references. Classes-Creating new data type in C++, class declaration, members, constructors and destructors, access functions, constant objects, member objects, static members, friend classes, arrays of class objects. Dynamic memory allocation-New and delete operators, class with pointer members, this pointer assignment, initialization, copy constructor, passing and returning objects, advanced free store techniques, exception handling. Inheritance and Polymorphism-Inheritance, Polymorphism: Operator overloading, handling related types in C++, derived class, conversion between base and derived classes, virtual functions, dynamic binding, pure virtual functions, protected members, public and private base classes, new, delete operators overloading, inheritance applications. Advanced C++ concepts-File handling, templates, container classes, class library, stack, queue and linked list applications, simple database applications.	
Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	1. **Introduction** • Comparison of procedure-oriented and structured oriented programming. • Comparison of event-driven, and object-oriented programming paradigms. • Fundamental of object oriented design.	**4 Hours**
Week2,3,4	2. **Features of object-oriented programming.** • C++ Fundamentals-Keywords. • Data types • Standard I/O streams. • Function prototypes. • C++ enhancements over C. • Default function parameters. • Inline functions, overloaded functions and reference. • Variables comparison between pointers and references.	**15 Hours**

Week 5,6	3. **Classes** • Creating new data type in C++, class declaration, members, constructors and destructors, access functions, constant objects, member objects, static members, friend classes, arrays of class objects. 4. **Dynamic memory allocation** • New and delete operators. • Class with pointer members. • This pointer assignment. • Initialization, copy constructor, passing and returning objects. • Advanced free store techniques, exception handling. Continuous Assessment I	**10 Hours**
Week7,8	5. **Inheritance and Polymorphism** • Inheritance, Polymorphism • Operator overloading, handling related types in C++. • Derived class, conversion between base and derived classes.	**10 hours**

Week9,10,11,12	6. **Advanced C++ concepts-** File handling, templates, container classes, class library, stack, queue and linked list applications, simple database applications. 7. Problems and Solutions with Functions • Focusing on functions created by the programmer • Appreciating the difference between the interface and the implementation • Writing and compiling C++ programs that include user-defined functions, provide flow control, and pass data by value and by reference 8. Control Structures • Scope of an identifier, lifetime of a variable, and value-returning functions •Completing the presentation of the C++control structures : switch-case, for loop, break, and continue • Collaborating, modifying, and compiling a C++ program • Modifying C++ source code to reflect good practice in variable scope 9. **Continuous Assessment II**	**20 Hours**
After Week 12	10. **Examinations**	

Recommended Reading Material

1. Kyle Loudon. (2003). C++ Pocket Reference. O'Reilly Media; 1st edition, ISBN-13 : 978-0596004965.
2. Stanley Lippman, Josée Lajoie, Barbara Moo. (2012). C++ Primer. Addison-Wesley Professional; 5th edition, **ISBN-13: 978-0321714114**
3. Scott Meyers. (2014). Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14. O'Reilly Media, Incorporated; 1st edition, ISBN-13 : 978-1491903995
4. Dale, N. et al (2002). Programming and Problem Solving with C++ (3rd ed). Jones and Bartlett Publishers. ISBN: 0763721034

19.4 200 Level Second Semester

CSC202 - Programming in Python (3 units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Programming in python
Year of Study	II
Course Code	CSC202
Credit Hours	3
Contact Hours	36
Pre-requisite(s)	CSC 101
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Mrs. Oluwafemi Temitope** **Mr. Musa Kunya**
Course Description	This course is designed to introduce the concept of OOP and general programming to non-Computer Science students.
Course Objectives	This course would enable the understanding of the following: 1. To understand the advantage of python as a programming language. 2. To identify the various data types in Python 3. To learn how to use the various control statements in python. 4. To learn how to handle exceptions in python. 5. To understand OOP concepts using python. 6. To learn how to design simple applications in python
Learning Outcomes	At the end of the course, students will be able to: 1. Develop a greater understanding of the issues involved in programming language design and implementation 2. Develop an in-depth understanding of functional, logic, and object-oriented programming paradigms using Python

	3. Understand design/implementation issues involved with variable allocation and binding, control flow, types, subroutines, parameter passing 4. Introduce the underlining concept of the compilation process.
Teaching and Learning	Classes should be for 3hrs weekly. Introductory concepts are taught in the classroom Laboratory Sessions: Problem-solving learning approach will be emphasized through hands-on practical sessions which will be tailored towards solving real-life problems using any of the Python IDEs (PyCharm, Anaconda, etc).
Detailed Course Content	Python Basics, definition of programming terms, data types -Numerical types, Containers, Assignment operator, Control Flow, if/elif/else, for/range, while/break/continue, Conditional Expressions, Advanced iteration. Defining functions -Function definition, Return statement, Parameters, Passing by value, Global variables, local variable. Docstrings, Comments, Good programming practice. Methods. Reusing code: scripts and modules, importing objects from modules, Creating modules, '__main__' and module loading. Input and Output -Iterating over a file, operating system functionality: high-level file operations, Pattern matching on files. Exception handling in Python -Exceptions, Catching exceptions, Raising exceptions. Object-oriented programming (OOP) concepts.

Course Content Sequencing

Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	11. **Introduction** • Definition of terms • History of python • Python Applications	**3 Hours**
Week2-3	12. Setting up python interpreter 13. Data Types in Python	**6 Hours**

Week4-6	14. Control flow • Simple if • if-else • nested if • if-elif-else • for loop • while loop 15. Functions in python 16. Continuous Assessment I	**9 Hours**
Week7-8	17. Files I/O 18. Exception handling	**6 hours**
Week9,10,11,12	19. Variables 20. Comments 21. OOP in python 22. Continuous Assessment II	**12 Hours**
After Week 12	23. Examinations	

Recommended Reading Material

1. Mark Summerfield. (2010). Programming in Python 3: A Complete Introduction to the Python Language, Second Edition. ISBN 978-0-321-68056-3
2. Brian Heinold. (2012). A practical introduction to python programming. Department of Mathematics and Computer science, Mount St. Mary's University, Maryland, USA.

CSC204 – Computer Organisation and Assembly Language (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Computer Organization and Assembly Language
Year of Study	3
Course Code	CSC204
Credit Hours	3
Contact Hours	42
Pre-requisite(s)	CSC101
Mode of Delivery	Classroom Lectures and Laboratory Practical

Mode of Assessment	**Weight%**
Presentation	20%
Continuous Assessment	20%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Mr. Muhammad Ahmad Shehu**
Course Description	Introduce learners to how computers are organized and programmed at different abstraction levels and covers a wide range of topics in both computer hardware organization and programming using the assembly language paradigm.
Course Objectives	This course would enable the understanding of the following: 1. Identify the relationships between a hardware specification and the associated instruction set. 2. Identify the major components of CISC and RISC architectures, and explain their purposes and interactions. 3. Simulate the internal representation of data, and show how data is stored and accessed in memory 4. Demonstrate the Instruction Execution Cycle for the intel x86 5. Differentiate between high-level, assembly, and machine languages and write well-modularized computer programs in an assembly language, implementing decision, repetition, and procedures. 6. Use a debugger, and explain register contents 7. Demonstrate the use of special purpose registers, and show how the system stack is used for procedure calls and parameter passing. 8. Explain various mechanisms for implementing parallelism in hardware/software
Learning Outcomes	At the end of the course, students will be able to: 23. Understand why Computer organization and Assembly language programming is an important area in Computer science 24. Understand the flow of interaction between computer and the user 25. Understand computer system components relation towards solving user problem 26. Understand assembly language programming
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures, Recitations, Tutorials and Practical Sessions.

Detailed Course Content	Introduction: Data representation, and number bases, Fixed and Floating point systems, Boolean algebra - Manipulation and minimization of completely and incompletely specified Boolean functions. Physical properties of gates: fan-in, fan-out, propagation delay, timing diagrams and tri-state drivers. Combinational circuit analysis and design, basic, flip-flop, clocking and timing diagrams Registers, counters, RAMs, ROMs, PLAs, PLDs, and FPGAs. Switching theory. Internal representation of data, instructions, and addresses. Registers. CISC and RISC architectures. Instruction set, Micro-programs (systems of Micro-operations). Instruction Execution. Procedure call and return. Write well-modularized computer programs in Assembly Language. The relationships between H/L languages and the Computer Architecture that underlies their implementation: basic machine architecture assembles specification and translation of Programming Language Block. Implementing decision, repetition, and procedures. Debugging assembly language using a debugger. Editors, assemblers and linkers. Parallelism in hardware/software. Structured Languages, parameter passing mechanisms.

Course Content Sequencing

Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	Introduction: Data representation, and number bases, Fixed and Floating point systems	**3 Hours**
Week2, 3 and 4	Physical properties of gates: fan-in, fan-out, propagation delay, timing diagrams and tri-state drivers. Combinational circuit analysis and design, basic, flip-flop, clocking and timing diagrams Registers, counters, RAMs, ROMs, PLAs, PLDs, and FPGAs	**9 Hours**
Week5, 6 and 7	Switching theory. Internal representation of data, instructions, and addresses. Registers. CISC and RISC architectures. Instruction set, Micro-programs (systems of Micro-operations). Instruction Execution. Procedure call and return.	**9 Hours**
Week8 and 9	Write well-modularized computer programs in Assembly Language.	**6 hours**
Week10, 11 and 12	The relationships between H/L languages and the Computer Architecture that underlies their implementation: basic machine architecture assembles specification and translation of Programming Language Block. Implementing decision, repetition, and procedures	**9 Hours**
Week13 and 14	Debugging assembly language using a debugger. Editors, assemblers and linkers. Parallelism in hardware/software. Structured Languages, parameter passing mechanisms.	**6 Hours**

Recommended Reading Material

1. Carl Hammacher,"Computer Organization, "Fifth Edition, McGrawHill International Edition,2002
2. P.Pal Chaudhuri,"Computer Organization and Design",2nd Edition, PHI, 2003
3. William Stallings, "Computer organization and Architecture-Designing for Performance PHI, 2004
4. Kann, Charles W., "Introduction To MIPS Assembly Language Programming" (2015). Gettysburg College Open Educational Resources.
5. Assembly Language for Intel-Based Computers, 5th Edition, by Kip Irvine, Prentice-Hall, 2006 ,, http://www.asmirvine.com

CSC206 - Human Computer Interaction (2 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	**Human Computer Interaction [HCI]**
Year of Study	II
Course Code	CSC206
Credit Hours	2
Contact Hours	24
Pre-requisite(s)	CSC101
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Mr. Ihinkalu Olalekan Ebenezer** **Mr. Dauda Isiaka -Laboratory Instructor**
Course Description	Human–computer Interaction (HCI) is the study of interaction between people (users) and computers. Interaction between users and computers occurs at the user interface which includes both software and hardware.
Course Objectives	This course would enable the understanding of the following: 7. The concept of design of Usable and Functional interfaces that will enhance greatly, the interaction between Humans and Computers. 8. The abilities to develop/design useful interfaces, software and computer based systems

	9. Conduct user studies using common HCI methods, including observation, participatory design, and interviews. 10. Communicate findings through written reports and common HCI summations, including personas and scenarios. 11. Design and conduct usability tests for a product or service. 12. Create, justify, and critique interface designs by referencing design principles, design patterns, and theoretical frameworks. 13. Create prototypes that demonstrate the interactivity of user interfaces, web applications and other interactive systems. 14. Create design documents, such as wireframes, user flow models, and site maps. 15. Implement interactive interfaces in Python Programming environment
Learning Outcomes	At the end of the course, students will be able to: 27. Expose the students to the concept of design of Usable and Functional interfaces that will enhance greatly, the interaction between Humans and Computers. 28. Equip students with abilities to develop/design useful interfaces, software and computer based systems 29. Conduct user studies using common HCI methods, including observation, participatory design, and interviews. 30. Communicate findings through written reports and common HCI summations, including personas and scenarios. 31. Design and conduct usability tests for a product or service. 32. Create, justify, and critique interface designs by referencing design principles, design patterns, and theoretical frameworks. 33. Create prototypes that demonstrate the interactivity of user interfaces, web applications and other interactive systems. 34. Create design documents, such as wireframes, user flow models, and site maps. 35. Implement interactive interfaces in Python Programming environment
Teaching and Learning	The class will meet for two hours each week. Class time will be used for a combination of Lectures, Seminar Presentation, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving and software modelling using Python
Detailed Course Content	Introduction and Course Overview: Definition of Human Computer Interaction, Overview of Human Computer Interaction, The goals of HCI Studies: Interaction technique, Interaction styles, Paradigms and History, Paradigms of interaction, Basic Definitions and Terminologies. Basic components of human computer interaction. The Interaction Models: Ergonomics, Common Interaction styles, Context: Social and Organisational, Common Architectures for design of HCI systems, Unimodal Systems, Visual Based HCI, Audio Based HCI, Sensor, Based HCI, Multimodal Systems, HCI Models, User Activity levels, Physical levels, Cognitive levels, Affective levels, Design Techniques-Task Analysis, Prototyping, User Testing, Evaluation Methods, Cognitive Walkthrough, Heuristic Evaluation, Review-based evaluation, Evaluating through user Participation, Design guidelines, rules and principles, Participatory design. Responsible Research, Design thinking, User Experience (Ux) Design: tools for Ux –Journey Maps, Story Boards, Wire Mesh, etc.
Course Content Sequencing	

Weeks	Detailed Course Outline	Allocated Time
Week1	• Introduction and Course Overview: Definition of Human Computer Interaction, Overview of Human Computer Interaction	**2 Hours**
Week2,3,4	• The goals of HCI: Interaction technique, Interaction styles, Paradigms and History, Paradigms of interaction, Basic Definitions and Terminologies. 1. Continuous Assessment I	**6 Hours**
Week5,6	• Basic components of Human Computer Interaction. • Cognitive Models • Design Process	**4 Hours**
Week7,8	• Interaction Design Basics • Software Process • Sensors mechanism 2. Seminar Presentation On each Sub – Topics of HCI, the students from the beginning of the semester will be shared in groups to present seminar topics and graded appropriately according to their group performance.	**4 hours**
Week9,10,11,12	• Universal Design • Implementation Support/Tools • Introduction to Python Programming 3. Continuous Assessment II	**8 Hours**
After Week 12	4. Examinations	

Recommended Reading Material

1. Dix, A., Finlay, J., Abowd, G.D., & Beale, R. (2004). Human computer interaction (3rd ed.). Prentice Hall. ISBN 0-13-046109-1.

2. You can find all of the resources related to this book online from the book's website at http://www.hcibook.com/e3/plain/about/book/. This is currently the major textbook used for teaching undergraduate HCI courses.

3. Preece, J., Rogers, Y., & Sharp, H. (2015). Interaction design: Beyond human-computer interaction (4th ed.) John Wiley & Sons Ltd. ISBN 978-1-119-02075-2.

4. You can find all of the resources related to this book online from the book's website at http://www.id-book.com/index.php.

5. Brad A. Myers. "A Brief History of Human Computer Interaction Technology." ACM interactions. Vol. 5, no. 2, March, 1998. pp. 44-54.

6. Hewett, Baecker etc., ACM SIGCHI Curricula for Human-Computer Interaction, http://old.sigchi.org/cdg/cdg2.html

CSC 208 - Introduction to Artificial Intelligence (3 Units)

FEDERAL UNIVERSITY LOKOJA **COURSE OUTLINE**	
Faculty	Sciences
Department	Computer Science
Course Title	Introduction to Artificial Intelligence
Year of Study	IV
Course Code	CSC208
Credit Hours	3
Contact Hours	36
Pre-requisite(s)	CSC 102
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Fati Oiza Ochepa (Mrs)** **Mr. Paulinus Umeh -Laboratory Instructor**
Course Description	In this course, the concepts, techniques, applications, and theories of Artificial Intelligence will be examined. Given the broad range of topics addressed by the AI field, topics for discussion must, necessarily, be limited. Therefore, this course will focus on issues of search, knowledge representation, reasoning, decision making, and learning from the perspective of an intelligent agent.
Course Objectives	This course would enable students have understanding of the following: 1. Provide students with knowledge of the history and revolution of artificial intelligence 2. Provide students with the understanding of the capabilities and the AI technologies. 3. Train the students in the process of administering search techniques 4. Inculcate in the students, the ability to design an application of artificial intelligence

	5. Formalize and implement constraints in search problems 6. Explore the concepts of planning, Producer -Consumer problems.
Learning Outcomes	At the end of the course, students will be able to: 36. To Analyse and apply search techniques; 37. To Analyse and apply adversarial search techniques; 38. Identify constraint satisfaction problems; 39. Describe, analyse and apply techniques for constraint satisfaction problems; 40. Describe and explain learning algorithms; 41. Design an application of artificial intelligence (AI); and 42. Write and present a demonstration of and a technical paper about the AI system designed. 43. Demonstrate working knowledge in a chosen AI language. 44. Represent knowledge in different knowledge representation patterns in Artificial Intelligence such as Frames and Pillars, semantic network, predicate logic/calculus, etc. 45. Describe and give examples of Producer -Consumer problems
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions be used to demonstrate working knowledge using Python
Detailed Course Content	Introduction to AI: History of AI, perspectives – philosophical, mathematical, psychological, etc., AI framework, Ai and associated technology, thinking vs. acting, humanly vs. rationally, knowledge of the Turing test and "Chinese Room". Intelligent Agents: Agents and environments, rationality, PEAS, environment and agent types. Search and constraint satisfaction: Formulation of problem spaces, knowledge of brute-force search methods (breadth-first, depth-first, iterative deepening), informed search methods (best first, A*), heuristics and admissibility, formulation and solution of constraint satisfaction problems, experience with adversarial search (minimax and alpha-beta pruning) Advanced search: Knowledge and experience with genetic algorithms, local search. Knowledge representation and reasoning: Frames, scripts, conceptual graphs, and conceptual dependency, Logical agents, propositional and first order logic, inference in first order logic (forward and backward chaining, resolution, theorem proving). Reasoning under uncertainty: Probabilistic reasoning, understanding of Bayesian networks and complexity of inference, temporal reasoning (smoothing, filtering and prediction), certainty factor. Planning: Definition and examples of Producer - Consumer problems-synchronous section-message sending by blocking or

	non-blocking-send/receive, CPU Scheduling-microprocessor scheduling.	
Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	**Introduction to AI** • History of AI • Perspectives – philosophical, mathematical, psychological, etc. • AI framework • AI and associated technology • Thinking vs. acting, humanly vs. rationally. • Knowledge of the Turing test and "Chinese Room".	**3 Hours**
Week2,3	**Intelligent Agents** • Agents and environments • Rationality. • PEAS • Environment and agent types.	**6 Hours**
Week4,5,6	**Search and Constraint Satisfaction** • Formulation of problem spaces • Knowledge of brute-force search methods (breadth-first, depth-first, iterative deepening). • Informed search methods (best first, A*) • Heuristics and admissibility • Formulation and solution of constraint satisfaction problems. • Experience with adversarial search (minimax and alpha-beta pruning). Continuous Assessment I	**9 Hours**
Week7,8	**Advanced search**. • Knowledge representation and reasoning: Frames, scripts, conceptual graphs, and conceptual dependency. • Logical agents, propositional and first order logic, inference in first order logic (forward and backward chaining, resolution, theorem proving). • Inference in Predicate Logic	**6 Hours**

Week9,10	**Reasoning under uncertainty** • Probabilistic reasoning, • Understanding of Bayesian networks and complexity of inference, temporal reasoning (smoothing, filtering and prediction) • Certainty factor. Continuous Assessment II	**6 Hours**
Week 11,12	**Planning** • Definition and examples of Producer-Consumer problems • Synchronous section-message sending by blocking or non-blocking-send/receive • CPU Scheduling-microprocessor scheduling.	**6 Hours**
After Week 12	**Examinations**	

Recommended Reading Material

1. Negnevitsky, M. (2005). Artificial Intelligence: A Guide to Intelligent Systems. Pearson Education Limited, Edinburgh Gate Harlow, Essex, England.
2. Russell S. J. and Norvig Peter. (1995). Artificial Intelligence A Modern Approach. Prentice Hall, Englewood Cliffs, NJ. Hopfield.
3. Nilsson, N.J.: "Artificial Intelligence, a new synthesis", Ed. Morgan Kaufmann Publishers, 1998
4. Bishop C.M.(2006) . Pattern Recognition and Machine Learning. Springer.
5. Duda R.O., Hart P.E. &Stork D.G. (2000). Pattern Classification; Wiley.
6. Haykin S. (1999). Neural Networks: A Comprehensive Foundation. Segunda edición. Prentice-Hall.
7. A.M. Turing. (1950). "Computing machinery and intelligence" Mind, Vol. 59, No. 236, pp. 433-460. http://www.jstor.org/pss/2251299

CSC 212 - Computer Programming II (2 Units)

FEDERAL UNIVERSITY LOKOJA **COURSE OUTLINE**	
Faculty	Sciences
Department	Computer Science
Course Title	Computer Programming II
Year of Study	II
Course Code	CSC
Credit Hours	2
Contact Hours	28
Pre-requisite(s)	CSC101

Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Abubakar Aliyu and Abdulwahab A. Jattau**
Course Description	This course introduces computer programming using JAVA programming language with object-oriented programming principles. Emphasis is placed on event-driven programming methods, including creating and manipulating objects, classes, and using object-oriented tools such as the class debugger. This course will prepare the students to all object-oriented programming course in future.
Course Objectives	At the end of the course, students will be able to: a) describe the computer system b) explain the concept of problem solving c) State the categories of various programming languages d) Explain the evaluation of major programming languages e) Design, create, build, and debug Java applications and applets. f) Implement syntax rules in Java programs. g) Explain variables and data types used in program development.
Learning Outcomes	Upon completion of this course, the student will be able to: a) Understand the basic concept of computer system b) Understand the concept of problem solving (apply algorithmic thinking to solve programming problems) c) Outline the categories of various programming languages d) Explain the evaluation of major programming languages e) Explain the historical development of Java Programming language f) Design, create, build, and debug Java applications and applets. g) Implement syntax rules in Java programs. h) Explain variables and data types used in program development. i) Apply arithmetic operations for displaying numeric output. j) Write and apply decision structures for determining different operations. k) Write and apply loop structures to perform repetitive tasks. l) Write user-defined methods. m) Identify and implement arrays, array lists, and multidimensional arrays. n) Write Java programs using object-oriented programming techniques including classes, objects, methods, instance variables, composition, inheritance, and polymorphism. o) Write programs using graphical user interface (GUI) components and Java's Event Handling Model. p) Develop custom classes using encapsulation, polymorphism, and abstraction.
Teaching and Learning	The class will meet for two hours each week. Class time will be used for a combination of Lectures, Recitations, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving and software development using Java programming language
Detailed Course Content	Overview of computer system. Concept of problem solving (applying algorithmic thinking to solve programming problems). Categories of various programming languages. Evaluation of major programming languages.

	Historical development of Java Programming language, Features of Java, how Java works, Types of Java Programs, Edit, compile, and run Java applications and applets. Variables, data types, and expressions- Identifier rules, Naming variables, constants (final) and references, Primitive data types, Arithmetic Operators, Assignment Operators, Relational and Logical Operators. Program control flow-Sequence structure, Selection structure, Repetition structure, Jump (Sequence) structure. Methods - Java API and Package/Library methods, User-defined methods, Scope and duration, Local and Field variables, Pass-by-value, Pass-by-reference, Recursion, Overloading. Arrays- Declaration and allocation, passing arrays to methods, sorting, searching Multiple-subscripted. Object-Based Programming-Classes and objects, instance variables, and instance methods, Member access modifiers: public, private, protected, package, Creating packages, Constructors, overloaded constructors, Set (mutator), Get (access), and predicate methods, Final instance variables, Composition, Finalizers, garbage collection, Static class members, This reference. Object-Oriented Programming – Inheritance, Superclass, subclass, Polymorphism, Dynamic method binding, abstract class, Concrete class, Inner class definition, Type-wrapper class for primitive data types, Interfaces. Graphical User Interface. Event-Driven Programming and Event Handling Model, Window Components, Mouse and keyboard event handling, Adapter classes, Layout managers	
Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**
Week 1	1. Overview of computer system. 2. Concept of problem solving I and II (applying algorithmic thinking to solve programming problems).	**2 Hours**
Week 2	3. categories of various programming languages. 4. evaluation of major programming languages. 5. historical development of Java Programming language 6. introduction to object-oriented programming	**2 Hours**
Week 3, 4, and 5.	7. First java program. 8. Features of Java and syntax of Java program. 9. Types of Java Programs, Edit, compile, and run Java applications and applets. 10. Variables, data types, and expressions- Identifier rules, Naming variables, constants (final) and references,	**9 Hours**
Week 6 and 7	11. Primitive data types 12. Arithmetic Operators 13. Assignment Operators 14. Relational and Logical Operators. 15. Program control flow-Sequence structure, 16. Selection structure 17. Repetition structure 18. Jump (Sequence) structure.	**6 hours**

Week 8, 9,10 and 11.	19. Methods - Java API and Package/Library methods, 20. User-defined methods, 21. Scope and duration, 22. Local and Field variables, 23. Parameter Passing (Pass-by-value and Pass-by-reference) 24. Recursion. 25. Overloading. 26. Arrays (Declaration and allocation, passing arrays to methods, sorting, searching Multiple-subscripted). 27. Object-Based Concept (Classes and objects, instance variables, and instance methods), 28. Member access modifiers: public, private, protected, package, 29. Creating packages, Constructors, overloaded constructors, Set (mutator), Get (access), and predicate methods, Final instance variables, Composition, Finalizers, garbage collection, Static class members, This reference. 30. Object-Oriented Programming – Inheritance, Superclass, subclass, Polymorphism, 31. Dynamic method binding, abstract class, Concrete class, Inner class definition, Type-wrapper class for primitive data types, Interfaces. 32. Graphical User Interface. Event-Driven Programming and Event Handling Model, Window Components, Mouse and keyboard event handling, 33. Adapter classes, Layout managers **Continuous Assessment**	**12 Hours**
After Week 12	34. Examinations	

Recommended Reading Material

1. Malik D. S. (2018) Java Programming: From Problem Analysis to Program Design, Fourth Edition, Pearson Edition
2. John Lewis, William Loftus, (2001) Java Software Solutions, Addison Wesley, 2001.
3. Bradley Kjell, Introduction to (2008) Computer Science using Java.
4. Paul J., Deitel, Tem R. Nieto (2012) Java How to program. 8th Edition. Pearson Education Limited.
5. Peter Nikolopoulos. (1997). Expert Systems: Introduction to First and Second Generation and Hybrid Knowledge Based Systems 1st Edition. ISBN-13: 978-0824799274

CSC222 Computer Electronics (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE		
Faculty	Sciences	
Department	Computer Science	
Course Title	**Computer Electronics**	
Year of Study	II	
Course Code	CSC222	
Credit Hours	3	
Contact Hours	36	
Pre-requisite(s)		
Mode of Delivery	Classroom Lectures Term Paper Presentations Laboratory Practical Sessions	
Mode of Assessment		**Weight%**

Continuous Assessment Final Examination	40% 60%
Total	**100%**

Course Lecturers and Instructor(s)	**Dr. Emeka Emmanuel Ogbuju** **-Laboratory Instructors**
Course Description	This course is to prepare students on how to develop innovative models and conduct comprehensive computer electronics simulations, to solve diverse real-life problems of our society. The aim of this course is to teach students the principles of computer electronics and the interface between computer science and electronics; how computer science and electronics can be used to implement 'real-life scenarios' in order to solve real-life problems.
Course Objectives	At the end of the course, students are expected to: *i)* Understand the meaning of computer electronics *ii)* Understand the applications of computer electronics *iii)* Understand the interface between computer science and electronics
Learning Outcomes	At the end of the course, students will be able to: 46. Do some simple practical implementations of some computer electronics projects such as Seven segment display 47. Perform practical mini projects on traffic light systems
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures, Group project Presentation, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving by major class groups using Assembly Language Programming, Proteus, etc.
Detailed Course Content	Introduction to Computer Electronics; what is Computer Electronics? Applications of Computer Electronics. Microcontroller Architecture; and Data to the LCD; Project 6E: To Introduction to Programmable chips; Brief History of Microcontrollers; Features of a Microcontroller; Types of Microcontrollers; The INTEL Microcontrollers; The ATMEL 89XX Series; THE AT89C52 Microcontroller (Description); The Microcontroller Memory; Register Banks. Basic Electricity; Component's description of electrical equipment. Assembly Language Programming; Features and Characteristics of Assembly Language; The 8051 Instruction Sets; More on Assembly Language Programming; Software Requirements for Chip Programming; Structure of Assembly Language Program for Microcontrollers; How to Burn (Transfer) Program into the Micro-Controller; Procedure for the Burning Process; Other Important Notes. Working with Light Emitting Diodes (LEDS). Working with KEYPADS; Working with MOTORS. Display Units; Seven Segment Display; Project 6A: To display numbers on the 7-segment display; Circuit description; Software Design; Program Codes for Project 6A ; Project 6B: To display static text on 7-segment display; Circuit description; Software Design; Program Codes for Project 6B. Project 6C: To display moving text on the Seven Segment Display; Circuit Description; Software Design; Program Codes for Project 6C. Project 6D: To use Keypad to control seven segment display; Circuit Description; Software Design; Program Codes for Project 6D. Liquid Crystal Display (LCD); LCD Connections and PinOuts; PIN Descriptions; LCD Memory; Basic Commands of a HD44780 LCD; Interfacing the LCD to a Microcontroller; Sending Command display the text on the LCD; Circuit Description; Software Design; Program Codes for Project 6E; Dot Matrix Display. Project 6F: Information Display on a Dot Matrix Display; Circuit Description; Software Design. Project 6G: Practical on 14-Segment Display - Simulations and hardware practical. Mathematical Operations with Microcontrollers; Introduction to Logic Gates, Counters; Decoders; Encoders; Electronic Workbench
Course Content Sequencing	

Weeks	Detailed Course Outline	Allocated Time
Week1	Introduction to Computer Electronics; What is Computer Electronics? Applications of Computer Electronics. Microcontroller Architecture; Introduction to Programmable chips; Brief History of Microcontrollers; Features of a Microcontroller; Types of Microcontrollers; The INTEL Microcontrollers; The ATMEL 89XX Series; THE AT89C52 Microcontroller (Description);	**3 Hours**
Week2,3,4	The Microcontroller Memory; Register Banks. Basic Electricity; Component's description of electrical equipment. Assembly Language Programming; Features and Characteristics of Assembly Language;	**9 Hours**
Week,5,6	The 8051 Instruction Sets; More on Assembly Language Programming; Software Requirements for Chip Programming; Structure of Assembly Language Program for Microcontrollers;	**6 Hours**
Week7,8	How to Burn (Transfer) Program into the Micro-Controller; Procedure for the Burning Process; Other Important Notes. Working with Light Emitting Diodes (LEDS). Working with KEYPADS; Working with MOTORS.	**6 hours**
Week9,10,11,12	**Laboratory Practical:** Display Units; Seven Segment Display; Project 6A: To display numbers on the 7-segment display; Circuit description; Software Design; Program Codes for Project 6A ; Project 6B: To display static text on 7-segment display; Circuit description; Software Design; Program Codes for Project 6B. Project 6C: To display moving text on the Seven Segment Display; Circuit Description; Software Design; Program Codes for Project 6C. Project 6D: To use Keypad to control seven segment display; Circuit Description; Software Design; Program Codes for Project 6D. Liquid Crystal Display (LCD); LCD Connections and PinOuts; PIN Descriptions; LCD Memory; Basic Commands of a HD44780 LCD; Interfacing the LCD to a Microcontroller; Sending Command display the text on the LCD; Circuit Description; Software Design; Program Codes for Project 6E; Dot Matrix Display. Project 6F: Information Display on a Dot Matrix Display; Circuit Description; Software Design. Project 6G: Practical on 14-Segment Display - Simulations and hardware practical. Mathematical Operations with Microcontrollers; Introduction to Logic Gates, Counters; Decoders; Encoders; Electronic Workbench **Project Implementations and Presentations** **Continuous Assessment II: Written Test**	**12 Hours**
After Week 12	Examinations	

Recommended Reading Material

1. V. Rajaraman, T. Radhakrishan, Introduction to Digital Computer Design 5th Edition, Prentice-Hall of India Pvt.Ltd 9788120334090
2. DC Tayal & Praveen Tayal, Fundamentals on Electronics 1st Edition, Morgan & Claypool 9781627055628
3. Introduction to Computer Electronics by Dozie Ekwunife

19.5 300 Level First Semester

CSC 301 - Structured Programming (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences

Department	Computer Science
Course Title	Structured Programming
Year of Study	III
Course Code	CSC 301
Credit Hours	3
Contact Hours	42
Pre-requisite(s)	CSC211
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturer	**Abubakar Aliyu**
Course Description	In this course the student will learn the basic Concept of Problem Solving using Structured Programming paradigm
Course Objectives	This course would enable the understanding of the following: 1. General overview of Computer System 2. Overview of programming Languages 3. General Concept of problem solving 4. Concept of programming Languages 5. Evaluation of major programming languages 6. Statement level Control structures 7. History and Concept of Structured Programming
Learning Outcomes	At the end of this course, students should be able to: 1. Understand the general concept of Computer System 2. Outline the basic categories of programming Languages 3. General Concept of problem solving 4. Understand the Concept of programming Languages 5. Evaluates the major programming languages 6. Understand Statement level Control structures 7. Understand the history and rational of structured programming languages. 8. Know the basic concepts of top- down analysis, modular programming and structured code. 9. Know the importance of studying structured programming languages. 10. Familiarize with some structured programming languages paradigms 11. Understand the fundamental concepts of structured design approaches such as, modular programming, top-down design bottom-up design and stepwise refinement. 12. Familiarize with the use of sub-programs. 13. Understand the fundamental concepts of control flow constructs in structured programming languages. 14. Familiarize with debugging and identify the various types of debugging techniques. 15. Understand the concepts of program testing and types of testing techniques
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures, Recitations, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving and software development using C, C++ or Python.
Detailed Course Content	Brief overview of Computer System. Overview of Programming Languages. General Concept of Problem Solving. Understand the Concept of

	Programming Languages. Evaluation of Major Programming Languages. Semantic Level Control Structures. Introduction to structured programming elements, structured design principles, Abstraction & modularity, Stepwise refinement, Introduction to structured design techniques, Comparison of structure-oriented programming with other contemporary paradigms, important advantages of structured programming over unstructured ones. Definition of structured programming approach like top-down analysis, abstraction, modular programming and structured coding. Structured programming languages and their advantages. Concepts of structured design approaches like modular programming, top-down design, bottom-up design stepwise refinement. Basics of data representation and manipulation, Function basics, Local variables, Parameters and arguments, Recursion, Module basics, Exceptions, Testing and Debugging, Sorting and Searching. File Processing: Text files processing, Database Connection and operations.

Course Content Sequencing

Weeks	Detailed Course Outline	Allocated Time
Week 1 and 2	24. General Overview of Computers System 25. Overview of Programming Languages 26. Introduction to Problem Solving I 27. Introduction to Problem Solving II 28. Concept of programming languages	**6 Hours**
Week 3 and 4	29. Evaluation of Major Programming Languages 30. Statement level Control structures 31. Continuous Assessment I	**6 Hours**
Week 5 and 6	32. Introduction to Structured Programming 33. concepts of top- down analysis, modular programming and structured code. 34. importance of studying structured programming languages. 35. structured programming languages paradigms	**6 Hours**
Week 7, 8 and 9	36. Fundamental concepts of structured design approaches such as, - modular programming, - top-down design - bottom-up design and - stepwise refinement. 37. Concept and development of sub-programs. 38. Fundamental concepts of control flow constructs in structured programming languages.	**9 hours**
Week 10,11 and 12	39. Concept of debugging and identify the various types of debugging techniques. 40. Concepts of program testing and types of testing techniques 41. Continuous Assessment II	**9 Hours**
After Week 13	42. Examinations	

Recommended Reading Material

1. Conway, R., & GRIES, D. (1973) *An introduction to programming: A structured approach.* Cambridge, Mass: Winthrop, 1973.
2. Dahl, O. J., Dukstra, E. W.. & Hoare, C. A. R. *Structured programming.* New York: Academic Press, 1972.
3. Dijkstra, E. W. Programming considered as a human activity. *Proceedings of IFlP Congress* 65. Washington, D. C: Spartan Books, 1965.
4. Dukstra, E. W. Goto Statement considered harmful. *Communications of the ACM.* 1968, 11, 147-148; 538; 541. "*
5. Dijkstra, E. W. Structured programming. In P. Naur & B. Randell (Eds.), *Software engineering techniques.* Brussels: NATO Scientific Affairs Division, 1969.
6. Malik D. S. (2018) C++ Programming. 8th Edition, Pearson Edition

7. Paul J., Deitel, Tem R. Nieto (2012) C++ How to program. 8th Edition. Pearson Education Limited. ISBN: 10:0-273-75276-6
8. Timothy B. D'Orazio (2004) Programmimg in C++: Lesson and Application. ISBN: 13:978-0-07-24-2412-6. McGrew-Hill
9. Rajiv Chopra. (2014). Software testing: A practical Approach.4th Edition. S. K. Kataria abd Sons Limited.

CSC 303 - Fundamental Data Structure (C++ or Java) (3 Units)

FEDERAL UNIVERSITY LOKOJA **COURSE OUTLINE**	
Faculty	Science
Department	Computer Science
Course Title	Fundamentals of Data Structures
Year of Study	3
Course Code	CSC 303
Credit Hours	3
Contact Hours	72
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	• **Malik Adeiza Rufai** • **Dauda O. Isiaka**
Course Description	Data structures are essential building blocks for designing efficient algorithms. Thus, they play a central role in computer science and are important in many areas of electrical engineering, computational biology, computational finance, etc. They are used in a variety of applications today including search engines (e.g., Google, Bing), social networking applications (e.g., Facebook, Twitter), embedded systems (e.g., cell phones, robots), and DNA analysis. This course will introduce the fundamentals of data structures and will provide a thorough understanding of how to systematically organize data in a computer system. In addition, this course will introduce students to analytical tools for comparing data structures in terms of their time and space complexities. Moreover, students will appreciate the importance of programming structures, abstractions, and algorithms for improving the efficiency of computer programs.

Course Objectives	At the end of the course, students should be able to: 1. Familiarize with good programming design methods, particularly Top-down design 2. Understand the Fundamental concepts of data, data structures, Abstract Data Type (ADT), algorithm and program. 3. Know how to choose a data structure type that will depend on the type of information involved, the Volume of data involved, the kind of manipulations to be done on the component of the data structure and how often processing is to be done on the components of the data structure. 2. Identify the attributes of data structures in order to determine the suitability of the data structure for particular applications. 3. Know how to store the components of a data structure in contiguous and linked storages. 4. Use arrays as a particular data structure type, its declarations, storage and manipulation within the computer system.
Learning Outcomes	At the end of the course, students will be able to: 1. Develop the data structures for linked lists, stacks, queues, trees and graphs. Implement the data structures using appropriate programming language (example, Python, Java, C/C++), in contiguous and linked storages. 2. Familiarize with the design and implementation of some simple searching, sorting algorithms. 3. Familiarize them self with memory management and hash tables. 4. Participate in programming Laboratory exercises to implement data structures covered in this course. 5. Familiarize with the use of pointers in storage (lined storage allocation) to obtain the following advantages: i) Prevent movement of components or interchange of positions during processing. ii) Gives rise to maximization of memory usage since excess memory declaration is eliminated iii) Allocation and re-allocation could be done while the program is running. iv) Save time because there is no interchange of component positions. v) Encourages efficient memory management.
Teaching and Learning	**Lectures:** The contents of the course should be presented to the students in the classroom, which should be in a form of lecture notes compiled from different relevant literature. Videos of lectures

	delivered by prominent scholars in data structures, can be shown to the students, so that they can have diverse ideas about the course. **Projects:** Individual and group projects will be given to the students. When group projects tasks are given, it is expected that the students should specialize in different sections of the project. The project should be a written program in C/C++ to solve problems in data structures. A printed copy of the results of each group project is submitted and serve as part of their continuous assessment scores.
Detailed Course Content	Introduction to some fundamental definitions of data, data structures, Abstract Data Type (ADT), algorithm and program, Attributes and criteria of choosing a data structure type. Memory organization and Data storage. Array: Definition, declaration and supplying the element of an array. Some operations on arrays, such as searching for a particular element, obtaining the highest element of an array and arranging the elements of the array in ascending order of magnitude. Linked List: The use of pointer and its advantages, Linked list illustration. Insert and deletion in a Linked list, Algorithm to find a component in linked list, Multi-linked list. Doubly and circular Linked list. Stacks and Queues: Stack data structure, Contiguous and linked Representations of a stack, Insertion and deletion in contiguous stack, Insertion and deletion in linked stack, Queue data structures, Queues in contiguous Representation, Circular Queue data structure, Linked Representation of Queues, Insertion and Deletion in Queues using linked Representation. Tree: Tree data structures, Binary tree Data structures and its applications, Binary tree traversal, Converting an Arbitrary tree to a Binary tree, Binary tree storage in contiguous and linked Representations. Graphs: Graph data structure, Applications of graphs, Adjacency matrix and Adjacency list Representations of graph, Graph Traversal. Searching and sorting methods: Linear and Binary Search methods, some sorting methods such as; Insertion, tree, selection, shell, merge and Quick sorting methods. Hashing: Definition of hashing, Hash Table representation, Hash functions such as division and multiplication methods, Problems in implementation of hashing such as collision and overflows, Collision Resolution such as Open Addressing and Chaining, Resolution of overflow, Records

	and sets in Data Structure.	
Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	43. **Data Structure Concepts** • Introduction to some fundamental definitions of data, • Data structures, Abstract Data Type (ADT), algorithm and program, Attributes and criteria of choosing a data structure type. • Memory organization and Data storage.	**6 Hours**
Week2,3	44. **Array and List** • Array: Definition, declaration and supplying the element of an array. Some operations on arrays, such as searching for a particular element, obtaining the highest element of an array and arranging the elements of the array in ascending order of magnitude. • Linked List: The use of pointer and its advantages, Linked list illustration. Insert and deletion in a Linked list, Algorithm to find a component in linked list, Multi-linked list. Doubly and circular Linked list. **Continuous Assessment I**	**12 Hours**
Week4,5,6	• Stacks and Queues: Stack data structure, Contiguous and linked Representations of a stack, Insertion and deletion in contiguous stack, Insertion and deletion in linked stack, • Queue data structures, Queues in contiguous Representation, Circular Queue data structure, Linked Representation of Queues, Insertion and Deletion in Queues using linked Representation.	**18 Hours**

Week7,8	• Tree: Tree data structures, Binary tree Data structures and its applications, Binary tree traversal, Converting an Arbitrary tree to a Binary tree, Binary tree storage in contiguous and linked Representations.	**12 hours**
Week9,10,11,12	• Hashing: Definition of hashing, Hash Table representation, Hash functions such as division and multiplication methods, • Problems in implementation of hashing such as collision and overflows, Collision Resolution such as Open Addressing and Chaining, Resolution of overflow, Records and sets in Data Structure. 45. **Continuous Assessment II**	**24 Hours**
After Week 12	46. Examinations	
Recommended Reading Material	Any available texts on Data Structures and Algorithms	

CSC305 - Compiler Construction I (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Compiler Construction I
Year of Study	III
Course Code	CSC305
Credit Hours	3
Contact Hours	36
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%

Total	**100%**
Course Lecturers	**Terungwa Simon Yange**
Course Description	The course gives an introduction to the design and implementation of a compiler with emphasis on principles and techniques for program analysis and translation. It also gives an overview of the tools for compiler construction. Lexical analysis, token selection, transition diagrams, and finite automata. The use of context-free grammars to describe syntax, derivations of parse trees, and construction of parsers. Syntax directed translation schemes; Intermediate code; Symbol table; Code generation; Detection, reporting, recovery and correction of errors.
Course Objectives	This course would enable the understanding of the following: 1. recognize various classes of grammars, languages, and automata, and employ these to solve common software problems 2. explain the major steps involved in compiling a high-level programming language down to a low-level target machine language 3. construct and use the major components of a modern compiler; 4. work together effectively in teams on a substantial software implementation project.
Learning Outcomes	At the end of the course, students will be able to: 1. Write and use simple object-oriented high-level programming languages to implement. sub-components/ sub-parts of a compiler 2. Use compiler construction tools to generate lexical and syntax analyzers. 3. Understand and be able to explain the functions of the different phases of a compiler. 4. Understand the key issues in the construction of production of compilers for real high-level Languages and real target machines 5. Understand how a compiler can generate code to make good use of some particular target machine characteristics
Teaching and Learning	1. **Lectures**: contents of the course will be presented and taught to students in the classroom. Classroom teachings will be supported with practical examples.

	2. **Projects**: Group and individual projects will be given to students to solve practical problems in compiler construction. Students will be expected to come to the classroom individually and defend their respective individual projects. 3. **Assignments**: Students will; be asked to solve class assignments with respect to topics covered in the class to examine, test the understanding of and reveal the state of assimilation of the course contents by students. 4. **Term Papers**: Students will be asked to write comprehensive term papers on selected sub-topics within the compiler construction course contents. The term papers will help students develop their understanding and in-depth analysis of components of the course contents. In some situations, students will be given a typical compiler construction research paper and will be asked to study, analyze and summarize in their own understanding, gaps and findings from the contents of the paper. Such term papers however will be subjected to thorough plagiarism checks.
Detailed Course Content	Introduction to compiler construction, Reasons why it is essential to study compiler construction, Languages and Translators, Types and role of grammars, Compiler structure and design issues, Phases of compiler, Internal Structure of a Compiler, Compile-time and run- time diagnostics , Symbol tables and their data structures, Lexical analysis, Token, Pattern and lexemes, Operations on languages, Regular expressions, Introduction to Syntax analysis ,Applications of Syntax analysis, Types of Errors encountered during compiler usage and how to recover from such errors. Introduction to and basics of Context-free Grammars, applications and examples. Functional relationship between lexical analysis, expression analysis and code generation. Use of a standard compiler as a working vehicle.

Course Content Sequencing

Weeks	Detailed Course Outline	Allocated Time
Week1	1. Introduction to compiler construction, Reasons why it is essential to study compiler construction, Languages and Translators	**3 Hours**

Week2,3	2. Types and role of grammars, Compiler structure and design issues, Phases of compiler, Internal Structure of a Compiler, Compile-time and run- time diagnostics **Continuous Assessment I**	**6 Hours**
Week4,5,6	3. Symbol tables and their data structures, Lexical analysis, Token, Pattern and lexemes, Operations on languages, Regular expressions, Introduction to Syntax analysis and Applications of Syntax analysis.	**9 Hours**
Week7,8	4. Types of Errors encountered during compiler usage and how to recover from such errors	**6 hours**
Week9,10,11,12	5. Introduction to and basics of Context-free Grammars, applications and examples. Functional relationship between lexical analysis, expression analysis and code generation. Use of a standard compiler as a working vehicle. 6. Revision and Continuous Assessment II	**12 Hours**
After Week 12	7. Examinations	

Recommended Reading Material

1. Aho, Alfred & Sethi, Ravi & Ullman, Jeffrey. *Compilers: Principles, Techniques, and Tools* ISBN 0201100886 The Classic Dragon book.
2. Appel*, A., Modern Compiler Implementation in Java,* 2nd ed., Cambridge University Press, 2002.
3. Appel, Andrew *Modern Compiler Implementation in C/Java/ML* (respectively ISBN 0-521-58390-X,ISBN 0-521-58388-8, ISBN 0-521-58274-1) is a set of cleanly written texts on compiler design, studied from various different methodological perspectives.
4. Brown, P.J. *Writing Interactive Compilers and Interpreters* ISBN 047127609X Useful practical advice, not much theory.
5. Fischer, Charles & LeBlanc, Richard. *Crafting A Compiler* ISBN 0805332014 Uses an ADA like pseudo-code.
6. Fischer, LeBlanc, Cytron*, Crafting a Compiler Implementation*, Addison-Wesley
7. Holub, Allen *Compiler Design in C* ISBN 0131550454 Extensive examples in "C".
8. Hunter, R. *The Design and Construction of Compilers* ISBN 0471280542 several chapters on theory of syntax analysis, plus discussion of parsing difficulties caused by features of various source languages.
9. Keith, D. Cooper & Linda Torczon, "Engineering a Compiler", Morgan Kaufmann Publishers, 2004
10. Pemberton, S. & Daniels, M.C. *Pascal Implementation. The P4 Compiler* ISBN 0853123586 (Discussion) and ISBN 085312437X (Compiler listing) Complete listing and readable commentary for a Pascal compiler written in Pascal.

11. Randy Allen and Ken Kennedy, "Optimising Compilers for Modern Architectures", Morgan Kaufmann Publishers, 2001.
12. Weinberg, G.M. *The Psychology of Computer Programming: Silver Anniversary Edition* ISBN 0932633420 Interesting insights and anecdotes.
13. Wirth, Niklaus *Compiler Construction* ISBN 0201403536 From the inventor of Pascal, Modula-2 and Oberon-2, examples in Oberon.

CSC307 - Database Management I (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Database Management I
Year of Study	III
Course Code	CSC307
Credit Hours	3
Contact Hours	36
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers	**Terungwa Simon Yange**
Course Description	The course is designed to provide students with a strong foundation in systematic approaches to design and implementation of database applications. Preliminarily operations like requirements gathering and database planning will be covered. The course will also introduce students to developing of application programs that talk to the database. These applications may be online or off line.
Course Objectives	This course would enable the understanding of the following: 1. Basic concepts pertaining to the data, information and knowledge management; 2. Demonstrate the variety of common contexts in which data organization and management is required in modern societies;

	3. Basic database concepts including the structure and operation of the relational data model 4. The functions of a DBMS and database administration as well as the role of a database management system in an organization 5. The structure and methods of the relational data model, and transform a logical data model into a relational database structure. 6. How to create and manipulate relational databases using QBE and SQL including constructing simple and moderately advanced database queries using Structured Query Language (SQL). 7. How to model logical data requirements using entity-oriented techniques 8. Apply normalization techniques to a database 9. Apply logical database design principles, including E-R diagrams and database normalization. 10. How to develop database applications using the most current database technologies
Learning Outcomes	At the end of the course, students will be able to: 1. Understand basic database concepts including the structure and operation of the relational data model 2. Understand the functions of a DBMS and database administration as well as the role of a database management system in an organization 3. Understand the structure and methods of the relational data model, and transform a logical data model into a relational database structure. 4. Create and manipulate relational databases using QBE and SQL including constructing simple and moderately advanced database queries using Structured Query Language (SQL). 5. Model logical data requirements using entity-oriented techniques 6. Apply normalization techniques to a database 7. Understand and successfully apply logical database design principles, including E-R diagrams and database normalization. 8. Develop database applications using the most current database technologies
Teaching and Learning	1. **Lectures:** Detailed content of course are taught in class using problem solving approaches 2. **Presentations:** Course contents are shared among students to research on. Students are grouped or assigned work to individually present. This is done for the purpose of self-reading improvement and student assessment.

	3. **Laboratory Practical:** Laboratory sessions will be based on the use of SQL to define and manipulate databases us any of MySQL, Oracle, MS Access, MSSQL *etc*. 4. **Project:** Students will be required to formulate problems in the different areas of the course content, design the solutions to these problems and solve them.
Detailed Course Content	Database Management Concepts: Background and history of databases, the general concepts of data management, basic database terminology, comparison between the file management and database management approaches, advantages and disadvantages of database processing, the Relational model. Database Design: entity-relationship modeling, user information analysis and representation, form table (first normal form (1NF), second normal form (2NF), third normal form (3NF) and fourth normal form (4NF)). Query by Example (QBE): Manipulating tables using QBE, use conditions, calculated fields, and built-in functions. Define multiple criteria, Use the logical operators Run action queries (make table, update, delete, and append). **Structured Query Language (SQL)**: Create tables, Insert data into a table, Run system catalog commands, Understand the format of the SQL SELECT command, Write single table queries, Write multiple table queries, Use simple conditions in queries, Use compound conditions in queries, Use computed columns in queries, Use the SUM, COUNT, MAX, MIN and AVG aggregate functions in queries, Use a subquery, Perform union, intersection and difference set operations in queries, Create views, Apply some of the administrative commands (Grant, Revoke, and Index). Practical aspects and demonstration of information storage & retrieval, information management applications, information capture and representation, analysis & indexing, search, retrieval, information privacy, integrity, security; scalability, efficiency and effectiveness. Introduction to advanced database topics, such as distributed database systems and the data warehouse. Relational Database: Mapping Conceptual schema to relational Schema; Database Query Languages (SQL). Concept of Fundamental dependencies & Multi-Valued dependencies. Transaction Processing; Distributed databases. Database Security. Internet Databases, Introduction to Data Warehousing and Data Mining. Database Recovery Techniques as well as emerging Database technologies. Several software libraries and publicly available data sets will be used to illustrate the application of selected algorithms, Emphasis will be on machine learning algorithms and applications.
Course Content Sequencing	

Weeks	Detailed Course Outline	Allocated Time
Week1	47. **Database Management Concepts** • Background and history of databases • General concepts of data management • Basic database terminology, • Comparison between the file management and database management approaches • Advantages and disadvantages of database processing • Relational model.	**3 Hours**
Week2,3	48. **Database Design and Query by Example** • Entity-relationship modeling • User information analysis and representation • First normal form (1NF) • Second normal form (2NF) • Third normal form (3NF) • Fourth normal form (4NF)) • Manipulating tables using QBE • Use conditions, calculated fields, and built-in functions • Define multiple criteria, Use the logical operators Run action queries (make table, update, delete, and append) Continuous Assessment I	**6 Hours**

Week4,5,6	49. Structured Query Language (SQL) • Create tables, insert data into a table, run system catalog commands • Understand the format of the SQL SELECT command • Write single and multiple table queries • Use simple and compound conditions in queries • Use computed columns in queries • Use the SUM, COUNT, MAX, MIN and AVG aggregate functions in queries • Use a subquery, perform union, intersection and difference set operations in queries • Create views, Apply some of the administrative commands (Grant, Revoke, and Index).	**9 Hours**
Week7,8	50. Practical aspects and demonstration of information storage & retrieval, information management applications, information capture and representation, analysis & indexing, search, retrieval, information privacy, integrity, security; scalability, efficiency and effectiveness.	**6 hours**
Week9,10,11,12	51. Introduction to advanced database topics, such as distributed database systems and the data warehouse. Relational Database: Mapping Conceptual schema to relational Schema; Database Query Languages (SQL). Concept of Fundamental dependencies & Multi-Valued dependencies. Transaction Processing; Distributed databases. Database Security. Internet Databases, Introduction to Data Warehousing and Data Mining. Database Recovery Techniques as well as emerging Database technologies. Several software libraries and publicly available data sets will be used to illustrate the application of selected algorithms, Emphasis will be on machine learning algorithms and applications. **Revision and Continuous Assessment II**	**12 Hours**
After Week 12	52. Examinations	
Recommended Reading Material		

1. Atzeni, P., Ceri, S., Paraboschi, S., & Torlone, R. (1999). *Database systems: concepts, languages & architectures* (Vol. 1). London: McGraw-Hill.
2. Batini, C., Ceri, S., & Navathe, S. B. (1992). *Conceptual database design: an Entityrelationship approach* (Vol. 116). Redwood City, CA: Benjamin/Cummings.
3. Dependencies‖,TODS, 1:4, December 1976.
4. Codd, E (1970) ―A Relational Model for Large Shared Data BANks‖ CACM, 136, June 1970.
5. Connolly, T. M., & Begg, C. E. (2005). *Database systems: a practical approach to design, implementation, and management*. Pearson Education
6. David M. Kroenke, David J. Auer (2008). Database Concepts. New Jersey. Prentice Hall
7. Elmasri Navathe (2003). Fundamentals of Database Systems. England. Addison Wesley.
8. Fred R. McFadden, Jeffrey A. Hoffer (1994). Modern Database management. England. Addison Wesley Longman
9. Graeme C. Simsion, Graham C. Witt (2004). Data Modeling Essentials. San Francisco. Morgan Kaufmann
10. Kim, W., Reiner, D. S., & Batory, D. (Eds.). (2012). *Query processing in database systems*. Springer Science & Business Media.
11. McHugh, J., Abiteboul, S., Goldman, R., Quass, D., & Widom, J. (1997). Lore: A database management system for semistructured data. *SIGMOD record*, *26*(3), 54-66.
12. Navathe, S. B., Tanaka, A. K., & Chakravarthy, S. (1992). Active Database Modeling and Design Tools: Issues, Approache, and Architecture. *IEEE Data Eng.*
13. *Bull.*, *15*(1-4), 6-9.
14. Pratt Adamski, Philip J. Pratt (2007). Concepts of Database Management. United States. Course Technology.
15. Singh, S. K. (2011). *Database systems: Concepts, design and applications*. Pearson Education India.
16. Teorey, T. J., Lightstone, S. S., Nadeau, T., & Jagadish, H. V. (2011). *Database modeling and design: logical design*. Elsevier.

CSC 311 - Algorithm and Complexity Analysis (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Algorithm and Complexity Analysis
Year of Study	3
Course Code	CSC311

Credit Hours	3
Contact Hours	42
Pre-requisite(s)	CSC203
Mode of Delivery	Classroom Lectures

Mode of Assessment	**Weight%**
Presentation	20%
Continuous Assessment	20%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Mr. Muhammad Ahmad Shehu**
Course Description	In this course the student will learn the approaches of measuring the degree of difficulties of algorithms for the purpose of identification of most suitable algorithm for a problem
Course Objectives	This course would enable the understanding of the following: 1. Know the big O, omega, and theta notations and their usage to give asymptotic upper, lower, and tight bounds on time and space complexity of algorithms. 2. Know how to determine the worst time complexity of algorithms Know how to deduce the recurrence relations that describe the time complexity of recursively-defined algorithms, and solve recurrence relations using mathematical induction and the recursion-tree method. 3. Design efficient algorithms and compare competing designs based on their complexities. 4. Be familiar with mathematical and scientific principles relevant to computer science. 5. Explain the mathematical concepts used in describing the complexity of an algorithm. 6. Select and apply algorithms appropriate to a particular situation. 7. Demonstrate basic understanding of some design approaches such as greedy algorithms, dynamic programming and divide-and-conquer. 8. Employ one from a range of strategies leading to the design of algorithms to serve particular purposes. 9. Explain the trade-offs that exist between a range of algorithms that possess the same functionality. 10. Be familiar with advanced and modern topics in computer science.

	11. Be able to debug implemented software in a proficient manner
Learning Outcomes	At the end of the course, students will be able to: 1. Understand why Algorithm and Complexity analysis is an important area in Computer science 2. Understand where and how the course can be applied 3. Identify the most suitable (in terms of efficiency) solution out of various solutions to a problem.
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures, Recitations, and Tutorials Sessions.
Detailed Course Content	Introduction to Algorithm Analysis, Fundamentals of Analysis of Algorithm Efficiency: Big Oh, Big Omega, and Big Theta notations. Rates of growth: worst, best, and average analysis, Design and Analysis of Searching and Sorting Algorithms: Optimal searching and sorting algorithms (linear search, binary search, Insertion sort, merge sort button-up sort, selection sort etc), Priority queues and heaps, heapsort, Quicksort, Sorting in linear time. Recurrence relations and their solution, Analysis of Divide-and-Conquer Algorithms: Analyzing recursive algorithms, Recurrence relations, Closest pair, Convex hull, Analysis of Decrease-and-Conquer Algorithms, Analysis of Transform-and-Conquer Algorithms, Analysis of Brute Force Algorithms, Space and Time Tradeoffs, Dynamic Programming, Design and Analysis of Graph Algorithms: Depth-first traversal algorithms, Breadth-first traversal algorithms, Minimum spanning tree, Shortest path algorithms, Advanced Designing Techniques: Dynamic Programming and Greedy Algorithms: Traveling salesperson approximation, Fibonacci numbers and binomial coefficients, All-pairs shortest path algorithm, Design and Analysis of Parallel Algorithms

Course Content Sequencing

Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	Introduction to Algorithm Analysis, Fundamentals of Analysis of Algorithm Efficiency: Big Oh, Big Omega, and Big Theta notations. Rates of growth: worst, best, and average analysis	**3 Hours**
Week2	Design and Analysis of Searching and Sorting Algorithms:	**3Hours**
Week3	Optimal searching and sorting algorithms (linear search, binary search, Insertion sort, merge sort button-up sort, selection sort etc)	**3 Hours**
Week4 and 5	Priority queues and heaps, heapsort, Quicksort, Sorting in linear time. Recurrence relations and their solution, Analysis of Divide-and-Conquer Algorithms: Analyzing recursive algorithms, Recurrence relations, Closest pair, Convex hull, Analysis of Decrease-and-Conquer Algorithms	**6 hours**

Week6 and 7	Analysis of Transform-and-Conquer Algorithms, Analysis of Brute Force Algorithms **Continuous Assignment (20marks)**	**6 Hours**
Week8, 9 and 10	Space and Time Tradeoffs, Dynamic Programming, Design and Analysis of Graph Algorithms: Depth-first traversal algorithms, Breadth-first traversal algorithms, Minimum spanning tree, Shortest path algorithms,	**9 Hours**
Week 11, 12, 13 and 14	Advanced Designing Techniques: Dynamic Programming and Greedy Algorithms: Traveling salesperson approximation, Fibonacci numbers and binomial coefficients, All-pairs shortest path algorithm, Design and Analysis of Parallel Algorithms	**12 Hours**

Recommended Reading Material

1. T. Cormen, C. Leiserson, and R. Rivest, Introduction to Algorithms, McGraw-Hill, New York, NY, 1992.
2. E. Horowitz, S. Sahni, and S. Rajasekaran, Fundamentals of Computer Algorithms, W. H. Freeman and Co., New York, NY, l998.
3. G. Rawlins, Compared to What: An Introduction to the Analysis of Algorithms, W. H. Freeman and Co., New York, NY, 1992.
4. S. Sahni, Data Structures, Algorithms, and Applications in Java, McGraw-Hill, NY, 2000.

CSC313 – Operations Research (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Operations Research
Year of Study	III
Course Code	CSC313
Credit Hours	3
Contact Hours	45
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment Final Examination	40% 60%
Total	**100%**

Course Lecturers and Instructor(s)	**Dr (Mrs) Taiwo Kolajo** **Laboratory Instructors**
Course Description	Operations research helps in solving problems in different environments that needs decisions. The course convers topics that include linear programming, Transportation, Assignment, and CPM/ MSPT techniques. Analytic techniques and computer packages will be used to solve problems facing

	business managers in decision environments. The course aims at improving students' problem-solving skill by subjecting them to real life problems and guide them through formulation of their solutions.
Course Objectives	This course would enable the understanding of the following: 1. To impart knowledge in concepts and tools of Operations Research 2. To understand mathematical models used in Operations Research 3. To use quantitative methods and techniques for effective decisions–making. 4. To apply these techniques constructively to make effective business decisions
Learning Outcomes	At the end of the course, students will be able to: 1. Understand the characteristics of different types of decision-making environments and the appropriate decision-making approaches and tools to be used in each type. 2. Solve Linear Programming Problems 3. Build and solve Transportation Models and Assignment Models. 4. Understand the usage of game theory and Simulation for Solving Business Problems. 5. Design new simple models, like: CPM, MSPT to improve decision –making and develop critical thinking and objective analysis of decision problems. 6. Implement practical cases by using TORA, WinQSB
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures, Recitations, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving using computer software packages like TORA, WinQSB, etc.
Detailed Course Content	Introduction to Operations Research: The Origin and Application of Operations Research, System Modelling Principles, Algorithm Efficiency and Complexity, Software for Operations Research. Linear Programming Problems (LLP): The LP Models, Graphical Solution of LPP, Simplex Method, Initial Solution for General Constraints, Information in the Tableau, Software for LPP, Illustrative Applications of LPP. Network Analysis: Graph and Networks - Preliminary Definitions, Maximum Flow in Networks, Minimum Cost Network Flow Problem, Network Connectivity, Shortest Path Problem, Dynamic Programming, Project Management, Software for Network Analysis, Illustrative Applications of Network Analysis. Integer Programming: Fundamental Concepts, Typical Integer Programming Problems, Zero–One (0–1) Model Formulations, Branch-and-Bound, Cutting Planes and Facets, Cover Inequalities, Lagrangian Relaxation, Column Generation, Software for Integer Programming, Illustrative Applications of Integer Programming. Queueing Models: Basic Elements of Queueing Systems, Arrival and Service Patterns, Analysis of Simple Queueing Systems, Software for Queueing Models, Illustrative Applications of Queueing Models

Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**

Week1	1. **Introduction to Operations Research** • The Origin and Application of Operations Research • System Modelling Principles • Algorithm Efficiency and Complexity • Software for Operations Research • Illustrative applications • Analytical Innovation in the Food and Agribusiness Industries, Humanitarian Relief in Natural Disasters, Mining and Social Conflicts	**3 Hours**

Week2,3,4	2. **Linear Programming Problems**
	• The Linear Programming Model
	  • Integer and Nonlinear Models
	• Graphical Solution of Linear Programming Problems
	  • General Definitions, Graphical Solutions, Multiple Optimal Solutions, No Optimal Solution, No Feasible Solution, General Solution Method
	• Simplex Method
	  • Standard Form of a Linear Programming Problem, Solutions of Linear Systems
	• Initial Solution for General Constraints
	  • Artificial Variables, The Two-Phase Method
	• Information in the Tableau
	  • Multiple Optimal Solutions, Unbounded Solution (No Optimal Solution), Degenerate Solutions, Analyzing the Optimal Tableau: Shadow Prices
	• Software for Linear Programming
	• Illustrative Applications
	  • Forest Pest Control Program, Aircraft and Munitions Procurement, Grape Processing: Materials Planning and Production
		12 Hours

Week5,6,7,8	3. Network Analysis
	• Graphs and Networks: Preliminary Definitions
	• Maximum Flow in Networks
	  • Maximum Flow Algorithm, Extensions to the Maximum Flow Problem
	• Minimum Cost Network Flow Problems
	  • Transportation Problem
	  • Northwest Corner Rule, Minimum Cost Method, Minimum "Row" Cost Method, Transportation Simplex Method
	  • Assignment Problem and Stable Matching
	  • Stable Matching
	  • Capacitated Transshipment Problem
	• Network Connectivity
	  • Minimum Spanning Trees, Shortest Network Problem: A Variation on Minimum Spanning Trees
	• Shortest Path Problems
	  • Shortest Path through an Acyclic Network, Shortest Paths from Source to All Other Nodes, Problems Solvable with Shortest Path Methods
	• Dynamic Programming
	  • Labeling Method for Multi-Stage Decision Making, Tabular Method, General Recursive Method .
	• Project Management
	  • Project Networks and Critical Paths, Cost versus Time Trade-Offs, Probabilistic Project Scheduling
	• Software for Network Analysis
	• Illustrative Applications
	  • DNA Sequence Comparison Using a Shortest Path Algorithm, Multiprocessor Network Traffic Scheduling, Shipping Cotton from Farms to Gins
	4. Continuous Assessment I
		15 Hours

Week9,10,11	5. Integer Programming
	• Fundamental Concepts
	• Typical Integer Programming Problems
	  • General Integer Problems, Zero–One (0–1) Problems, Mixed Integer Problems,
	• Zero–One (0–1) Model Formulations
	  • Traveling Salesman Model, Knapsack Model, Bin Packing Model, Set Partitioning/Covering/Packing Models, Generalized Assignment Model
	• Branch-and-Bound
	  • A Simple Example, A Basic Branch-and-Bound Algorithm, Knapsack Example
	  From Basic Method to Commercial Code
	  Branching Strategies
	  Bounding Strategies
	  Separation Rules
	  The Impact of Model Formulation
	  Representation of Real Numbers
	• Cutting Planes and Facets
	• Cover Inequalities
	• Lagrangian Relaxation
	  • Relaxing Integer Programming Constraints, A Simple Example, The Integrality Gap, The Generalized Assignment Problem, A Basic Lagrangian Relaxation Algorithm, A Customer Allocation Problem
	• Column Generation
	• Software for Integer Programming
	• Illustrative Applications
	  • Solid Waste Management, Timber Harvest Planning, Propane Bottling Plants
		10 hours

Week12	6. Queueing Models
	• Basic Elements of Queueing Systems
	• Arrival and Service Patterns
	  • The Exponential Distribution
	  • Birth-and-Death Processes
	• Analysis of Simple Queueing Systems
	  • Notation and Definitions
	  • Steady State Performance Measures
	  • Practical Limits of Queueing
	• Software for Queueing Models
	• Illustrative Applications
	  • Cost Efficiency and Service Quality in Hospitals
	  • Queueing Models in Manufacturing
	  • Nurse Staffing Based on Queueing Models
	7. Continuous Assessment II
		5 Hours

After Week12	1. Examination	

Recommended Reading Material

1. Hillier, Frederick & Lieberman, "Introduction to Operations Research Concepts and Cases", 2010, 8th Ed. TMH
2. N.D. Vohra, "Quantitative Techniques in Management", 2010, 4th Ed.TMH.
3. J.K. Sharma, "Operations Research Theory and Applications 2009,4th Ed. McMillan.
4. Kasana, HS & Kumar, KD, "Introductory Operations Research theory and Applications", 2008, Springer.
5. Chakravarty, P, "Quantitative Methods for Management and Economics", 2009, 1st Ed. HPH.
6. Murthy, P. R., "Operations Research", 2007, 2nd Ed. ISBN (13) : 978-81-224-2944-2

CSC315 - Computer Architecture and sequential programme (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Computer Architecture and sequential programme
Year of Study	III
Course Code	CSC315
Credit Hours	3
Contact Hours	36
Pre-requisite(s)	CSC 205
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Mrs Oluwafemi Temitope**
Course Description	Computer architecture is a specification detailing how a set of software and hardware technology standards interact to form a computer system or platform. In short, computer architecture refers to how a computer system is designed and what technologies it is compatible with. This course also deals with the performance evaluation of computer systems. The emphasis is on microprocessors, chip-multiprocessors and memory hierarchy design. Special attention is paid to pipelining, using hardware and/or software techniques to yield high performance. This course will benefit students to understand computer architecture concepts and mechanisms related to the design of modern processors, and memories and explain how these concepts and mechanisms interact.
Course Objectives	This course would enable the understanding of the following: 5. Describe computer architecture concepts and mechanisms related to the

	design of modern processors, and memories and explain how these concepts and mechanisms interact. 6. Apply this understanding to new computer architecture design problems within the context of balancing application requirements against technology constraints. 7. Understand the inner workings and performance of microprocessors. 8. Understand the concepts; registers, memory, addressing mode, physical memory address, assembler directive, program-controlled I/O, microprogrammed control. 9. Operate with concepts and notions related to floating point systems and operations. 10. Have an ability to evaluate hardware accelerator targeting at applications with substantial data-level parallelism. 11. Learn software-driven techniques to match application requirements to available pipelined hardware in order to obtain high performance. 12. Understand cache coherence issues. 13. Understand the basic of shared-memory
Learning Outcomes	At the end of the course, students will be able to: 1. Explain how computer hardware and software are being influenced by their architecture 2. Understand the computer from the programmers point and understand the overall structure and function of a computer. 3. Understand the concept of pipelining. 4. Select appropriate computer systems for given application domains.
Teaching and Learning	Classes should be for 3hrs weekly. Contents of the course will be presented and taught (in power point format) to students in the classroom. Classroom presentations will be supported with practical demonstrations. Students will equally participate in group presentations and tutorial sessions.
Detailed Course Content	Introduction to Digital System Architecture: Computer systems components, Brief historical background, Architectural development and Styles, Computer Organisation and design, Memory hierarchy, CISC Architecture: Von Neuman, computer organization, characteristics, merits, and demerits; RISC Architecture: Havard architecture, computer organization, characteristics, merits, and demerits, Examples of CISC Architecture: INTEL80x86, and Motorola 680xx microprocessor, Generations of Intel80x86

	microprocessor. Evolution of instruction sets, Characteristics and metrics of Instruction sets, Data Memory Storage formats: Little Endian vs Big Endian, 80x86 Programming models, 80x86 Addressing modes, M68000 Programming models, M68000 Addressing modes. Intel Processor Operation Modes: 80x86 Real Mode (characteristics, advantages, disadvantages, and operating systems), 80x86 Protected Mode (characteristics, advantages, disadvantages, and operating systems). 80x87 Floating Point Architecture: IEEE754 floating point standards, 80x87 floating point model, 80x87 Instruction Set, 80x87 floating point programming. Parallel Architecture: SIMD Architecture of 80x86, MMX architecture, SSE1-3 architectures, MMX /SSE Instruction sets. Pipelining: Basic and Intermediate Concepts.

Course Content Sequencing

Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	1. **Introduction** • Define computer Architecture • Explain the Functions of various parts of the computer • CPU Basics • Parts of the CPU	**3 Hours**
Week2-4	2. Memory Organization • Main memory • Cache • Instruction cycle 3. Computer Architecture • Von Neumann • Havard Architecture	**9 Hours**
Week5-6	4. Addressing Mode 5. CISC and RISC **6.** Continuous Assessment I	**6 Hours**
Week7-8	7. Intel 80x86 Architecture, 8. Motorolla 68000, Interrupts	**6 hours**
Week9,10,11,12	9. ISA 10. Parallel Arcitecture 11. Pipelinning 12. Continuous Assessment II	**12 Hours**
After Week 12	13. Examinations	

Recommended Reading Material

7. William Stallings. (2016). Computer Organization and architecture designing for performance. Tenth edition. ISBN 13:978-0-13-410161-3

CSC319 – Information Technology Law (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	**Information Technology Law**
Year of Study	III
Course Code	CSC319
Credit Hours	3
Contact Hours	36
Pre-requisite(s)	
Mode of Delivery	Classroom Lectures Term Paper Presentations (Group Work) Laboratory Practical Sessions
Mode of Assessment	**Weight%**
Continuous Assessment Final Examination	40% 60%
Total	**100%**
Course Lecturers and Instructor(s)	**Dr. Emeka Emmanuel Ogbuju -Laboratory Instructors**
Course Description	This course brings to fore the numerous challenges of Information Technology on the areas of cybercrimes, intellectual property, information and critical infrastructure security, professional ethics and IT policy legislation. It discusses the concepts, tools and techniques needed to combat the social menace caused by the misuse of the computer and internet technology. The course also covers the roles of government agencies both at the local and international levels to address the challenges. Though CSC319 is an elective course, the knowledge gleaned from participating in it will assist and equip students to stay safe and practice computing within relevant legal frameworks.
Course Objectives	By the end of this course, students are expected to have the following knowledge/skills: *a)* Practice computing with the requisite set of legal knowledge and ethics required in the discharge of their professional duties *b)* Acquire Information Technology security skills necessary for both offline and online activities *c)* Develop computer/software acquisition/development contracts *d)* Become national and international digital advocates for economic growth and development using the computer and internet technology
Learning Outcomes	At the end of the course, students will be able to:
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures, Group project Presentation, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions,

	while the Laboratory sessions will be based on problem-solving by major class groups using different ethical hacking tools.	
Detailed Course Content	Basic netiquettes. Introduction to computer ethics, Personal code of computer ethics and social values, Code of ethics and professional conduct (Nigeria context). Introduction to intellectual property protection: Copyright, patent, trademark, Introduction to digital signature and electronic signatures. IT Policy legislation, Computer contracts, Introduction to cybercrimes, categories of cybercrimes. International Cybercrime Initiatives: Budapest Convention on Cybercrime; Cybercrime initiatives of the D-8 countries, IT laws in China, Brazil, Russia, India, United Kingdom, and USA. Cybercrime survival tips (with lab sessions), Malware attacks, Scams. General discussion of the Nigerian Cybercrime (prohibition, prevention, detection, response, investigation and prosecution of cybercrimes; and for other related matters) Act 2015, National Cybesecurity Strategy. Online privacy and data protection, categories of privacy, privacy violation methods. Cyber security guide: the action phases, Introduction to Ethical Hacking (with lab sessions on Penetration Testing). Practical class/group work on development of a cyber-security framework for an information system. Presentation of Group Work. Tutorial and Revision for Course Examination	
Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	Basic netiquettes. Introduction to computer ethics, Personal code of computer ethics and social values	**3 Hours**
Week2,3,4	Code of ethics and professional conduct (Nigeria context). Introduction to intellectual property protection: Copyright, patent, trademark, Introduction to digital signature and electronic signatures. IT Policy legislation, Computer contracts, Introduction to cybercrimes, categories of cybercrimes. International Cybercrime Initiatives: Budapest Convention on Cybercrime; Cybercrime initiatives of the D-8 countries,	**9 Hours**
Week,5,6	IT laws in China, Brazil, Russia, India, United Kingdom, and USA. Cybercrime survival tips (with lab sessions), Malware attacks, Scams. General discussion of the Nigerian Cybercrime (prohibition, prevention, detection, response, investigation and prosecution of cybercrimes; and for other related matters) Act 2015, National Cybersecurity Strategy.	**6 Hours**
Week7,8	Online privacy and data protection, categories of privacy, privacy violation methods. Cyber security guide: the action phases, Introduction to Ethical Hacking (with lab sessions on Penetration Testing). Practical class/group work on development of a cyber-security framework for an information system.	**6 hours**
Week9,10,11,12	Presentation of Group Work. Tutorial and Revision for Course Examination **Project Implementations and Presentations Continuous Assessment II: Written Test**	**12 Hours**
After Week 12	Examinations	
Recommended Reading Material 1. Nigerian Cybercrime (prohibition, prevention, detection, response, investigation and prosecution of cybercrimes; and for other related matters) Act 2015 2. National Cybersecurity Strategy.		

CSC 321 - System Analysis and Design (3 Units)

COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	System Analysis and Design
Year of Study	III
Course Code	CSC321
Credit Hours	3
Contact Hours	36
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Fati Oiza Ochepa (Mrs)** **Mr. Paulinus Umeh -Laboratory Instructor**
Course Description	This course is a very important core course in computing profession which every practitioner will always use throughout the development of any system. The concepts, tools and techniques to be learnt will be a full part of student's computing career. We shall cover all the areas in planning, analysis, design and implementation of an information system.
Course Objectives	This course would enable the understanding of the following: 1. Understanding and application of the concepts of the system development lifecycle 2. Application of the analysis and design concepts to a system development problem 3. Application of the system development process flow using appropriate and standardized set of diagrams/drawings 4. Acquisition of relevant people and projects skills for the implementation of a system. 5. The process of transfer of human knowledge to a machine.
Learning Outcomes	At the end of the course, students will be able to: 7. Define Planning, Analysis and Design. 8. Select appropriate software development methodology. 9. Develop Use case analysis of practical problems.

	10. Apply the various categories of software development requirements. 11. Design and implement data flow diagrams. 12. Apply System Development Lifecycle in building a software project. 13. Develop a software, coding with C++, VB.Net, JAVA.
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures and Tutorials. Key concepts would be taught during instructor-led sessions. Lecture will be delivered using guided instructions and PowerPoint format and soft copies of lecture notes. There will be interactive classroom students' engagement sessions, Group and Individual Assignments/Tasks, Live Quizzes to assess the immediate students' understanding of concepts etc.
Detailed Course Content	Definitions and Introduction to Planning, Analysis, Design and Implementation. The Systems Analyst, Information Systems Development, System Development Lifecycle. Project Selection and Management, Project Methodology Options, Project success factor.Selecting the Appropriate Development Methodology, Critical systems. Use Case Analysis, Process Modeling, Data Modeling. Moving into Design, System Acquisition Strategies. Architecture Design, Operational Requirements, Performance Requirements, Security Requirements, Cultural and Political Requirements, Hardware/Software Specification. User Interface Design, Principles and Processes of User interface Design, HCI Issues, Navigation Design, Input/Output Design. Program Design, Physical Data Flow Diagram, Structure Chart, Program Specification. Data Storage Design. Implementation Phase, Coding with C++, VB.Net, JAVA. Practical System Analysis and Design I. Practical System Analysis and Design II. Tutorial and Revision for Course Examination

Course Content Sequencing

Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	• Definitions and Introduction to: • Planning, • Analysis • Design and • Implementation.	**3 Hours**

Week2,3,4	**2. The Systems Analyst.** • Information Systems Development • System Development Lifecycle. • Project Selection and Management, • Project Success factor • Project Methodology Options, • Selecting the Appropriate Development Methodology. • Critical Sytems • Use Case Analysis • Process Modeling, Data Modeling.	**9 Hours**
Week 5,6	**3. Design** • System Acquisition Strategies. • Architecture Design • Operational Requirements, Performance Requirements, Security Requirements, Cultural and Political Requirements • Hardware/Software Specification. **4.Continuous Assessment I**	**6 Hours**
Week 7,8	**5. User Interface Design** • Principles and Processes of User interface Design • HCI Issues, Navigation Design, Input/Output Design. • Program Design, Physical Data Flow Diagram, Structure Chart • Program Specification.	**6 Hours**
Week9,10	6. **Data Storage Design.** • Practical System Analysis and Design 7. Continuous Assessment II	**6 Hours**
Week 11, 12	8. Implementation Phase, Coding with C++, VB.Net, JAVA. Practical System Analysis and Design	**6 Hours**
After Week 12	9. **Examinations**	

Recommended Reading Material

1. Barry Williams. (2012). Data Modeling by Example. London. 1st Edition. ISBN-13: 978-1478114192
2. Ian Sommerville. (2004). Software Engineering. Addison-Wesley. 7th edition.
3. Hargitay S & Dixon T. (1991). Software Selection for Surveyors. 1st Edition.

19.6 300 Level Second Semester

CSC398 - SIWES (Industrials Training) (6 Units)
The students undego six (6) months industrial Training with Institutions or Organization relative to Computer Science discipline and as approved by the Departmental SIWES Coordinator.

19.7 400 Level First Semester

CSC401 - Software Development and engineering (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Software Development and engineering
Year of Study	IV
Course Code	CSC401
Credit Hours	3
Contact Hours	38
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Dr. Edgar. O. Osaghae**
Course Description	In this course the students will learn how to organize and manage a software development project successfully, and the students should combine specific knowledge, skills, efforts, experience, capabilities and the right intuition, to fully acquire the concepts.
Course Objectives	This course would enable the understanding of the following: 1. Provide students with the required skills in software Engineering. 2. Teach students the software process and project management skills. 3. Teach students the specifications, requirements analysis and software design process. 4. Teach students how to test and implement software products. 5. Teach students project management techniques.

Learning Outcomes	At the end of the course, students will be able to: 1. Understand the concepts of Software Engineering. 2. Know the artifacts to manage and control during software development. 3. Organize the software development team. 4. Know what are the indicators and measures of the software product's quality. 5. Employ a certain set of software development practices. 6. Translate a software development organization to a new modeling and/or development paradigm. 7. To create and maintain a good relationship with customers and end-users of their software products. 8. Remedial actions to take when something goes wrong in the course of the software engineering project.
Teaching and Learning	Lessons for this course would be conducted in two hours per week. Lesson time will comprise of classroom teachings and practical sessions. Most of the lesson time, will be spent on Instructor-led sessions, and the practical session would be Laboratory sessions and take-home assignment practical. The programming languages for the course are Java and Python.
Detailed Course Content	Software Process and Project Management; Introduction to Software Engineering, Software Process, Perspective and Specialized Process Models, Software Project Management: Estimation – LOC and FP Based Estimation, COCOMO Model, Project Scheduling, Scheduling, Earned Value Analysis - Risk Management. Requirements Analysis and Specification; Software Requirements: Functional and Non-Functional, User requirements, System requirements, Software Requirements Document and Requirement Engineering Process: Feasibility Studies, Requirements elicitation and analysis, requirements validation, requirements management-Classical analysis: Structured system Analysis, Petri Nets-Data Dictionary. Design Concepts, Design Model, Design Heuristic, Architectural Design; Design process Architectural styles, Architectural Design, Architectural Mapping using Data Flow- User Interface Design: Interface analysis, Interface Design– Component level Design: Designing Class based components, traditional Components. Testing and Implementation; Software Testing Fundamentals-Internal and External Views of Testing-White Box, Testing-Basis Path Testing Control Structure, Testing-Black Box, Testing-Regression Testing, Unit Testing, Integration Testing, Validation Testing, System Testing and Debugging, Software Implementation Techniques: Coding practices and Refactoring. Project Management; Estimation, FP Based, LOC Based,

	Make/Buy Decision, COCOMO II, Planning, Project Plan, Planning Process, RFP Risk Management, Identification, Projection, RMMM, Scheduling and Tracking, Relationship between people and effort, Task Set and Network, Scheduling, EVA, Process and Project Metrics.	
Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**
Week 1, 2	1. **Software Process and Project Management** • Introduction to Software Engineering • Software Process, Perspective and Specialized Process Models • Software Project management: Estimation • Requirements Analysis Design • Project Scheduling, Scheduling, Earned Value Analysis	**6 Hours**
Week 3,4	**2. Requirements Analysis and Specification** • Software Requirements: Functional and Non-Functional, User Requirements, System Requirements, Software Requirements Document • Requirement Engineering Process: Feasibility Studies, Requirements elicitation and analysis • Requirements Validation, Requirements Management • Classical Analysis • Structured Systems Analysis • Petri Net-Data Dictionary **Continuous Assessment I**	**7 Hours**

Week 5, 6, 7	3. **Software Design** • Design Process • Design Concepts-Design Model • Design Heuristic • Architectural Design • Architectural Styles, Architectural Design, Architectural Mapping using Data Flow • User Interface Design • Interface Analysis, Interface Design • Component Level Design • Designing Class Based Components, Traditional Components	**9 Hours**
Week 8, 9, 10	**4. Testing and Implementation** • Software Testing Fundamentals • Internal and External Views of Testing • White Box Testing-Basis Path Testing • Control Structure Testing • Black Box Testing**,** Regression Testing, Unit Testing, Integration Testing and Validation Testing • System Testing and Debugging • Software Implementation Techniques: Coding Practices Best Coding Practices • Refactoring	**9 hours**
Week 11, 12	**5. Project Management** • Estimation, FP Based, LOC Based, Make/Buy Decision, COCOMO II • Planning, Project Plan, Planning Process, RFP Risk Management • Identification, Projection, Risk Mitigation Monitoring and Management (RMMM) • Scheduling and Tracking • Relationship between People and Effort, Task Set and Network, Scheduling and Earned Value Analysis (EVA) **Continuous Assessment II**	**7 Hours**
After Week 12	Examinations	

Recommended Reading Material

8. Nico L. (2021). Software Engineering for Absolute Beginners, Springer Science + Business Media, New York, United States.
9. Ian S. (2011). Software Engineering, Addison-Wesley Publishing, Massachusetts, United States.
10. Roger S. P. & Bruce R. (2015). Software Engineering: Practitioner's Approach, McGraw-Hill Education, New York, United States.
11. Rajilch V. (2011). Software Engineering: The Current Practice, CRC Press, Taylor & Francis, Raton, United Kingdom.
12. Otero Carls E. (2012). Software Engineering Design: Theory and Practice, CRC Press, Taylor & Francis, Raton, United Kingdom.

CSC403 - Survey and Organization of Programming Languages (4 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Survey and Organization of Programming Languages
Year of Study	IV
Course Code	CSC403
Credit Hours	4
Contact Hours	39
Pre-requisite(s)	CSC 301
Mode of Delivery	Classroom Lectures Term Paper Presentations Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Dr. Victoria Ifeoluwa Yemi-Peters** **Laboratory Instructors**
Course Description	In this course Students will learn how programming languages work to broaden their language horizons on: • Different programming languages • Different language features and tradeoffs • Useful programming patterns • Study how languages are described / specified • Study how languages are implemented
Course Objectives	Students will be equipped with the knowledge of the following: That- 1. Programming languages vary in their • Syntax

	• Style/paradigm • Semantics • Semantics • Implementation 2. They are designed for different purposes • Goals changes as the computing landscape changes, e.g., as programmer time becomes more valuable than machine time. 3. Ideas from one language appear in others and also to know the appropriate programming language to solve a problem task efficiently and effectively.
Learning Outcomes	At the end of this course, students will be expected to be able to: Identify the distinctive characteristics of each of these three major language paradigms: o Procedural o Object-oriented o Functional • Write medium-sized programs in C++, JAVA,JAVA SCRIPT and PYTHON • Articulate the rationales for and differences between static and dynamic type systems • Evaluation by Comparing and contrasting the programming languages • Understand, apply, and distinguish parametric data types; techniques to include: o Inheritance o Parametric data types and type inference o Dynamic types o Algebraic data types o Type classes • Discuss the relative advantages and disadvantages of interpreters and compilers • Understand and apply the iteration techniques: • Discuss the trade-offs involved in programming with and without referential transparency. • Learn and explain a previously unknown programming language. • Contribute to a large project in a previously unknown programming language.
Teaching and Learning	The class will meet for Three or Four hours each week. Class time will be used for a combination of Lectures, Group work presentations, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, Interactive class discussions alongside with the group presentations, while the Laboratory sessions will be based on problem-solving and evaluations using the four programming languages mentioned above.
Detailed Course Content	Overview of programming languages: History of programming languages. Introduction to Collaborative programming- Pair programming paradigms, Egoless programming, Git Programming. Brief survey of programming paradigms (Procedural languages, Object-oriented languages, Functional languages,

	Declarative – non-algorithmic languages, Scripting languages), the effects of scale on programming methodology. Language Description: Syntactic Structure (Expression notations, abstract Syntax Tree, Lexical Syntax, Grammars for Expressions, Variants of Grammars), Language Semantics (Informal semantics, Overview of formal semantics, Denotation semantics, Axiomatic semantics, Operational semantics); Language definition structure. Data types and structures, Review of basic data types, including lists and tress, control structure and data flow, Run-time consideration, interpretative languages, lexical analysis and parsing; Declarations and types: The concept of types, Declaration models (binding, visibility, scope, and lifetime), Overview of type-checking; Memory Management: Activation records and Garbage collection; Data Abstraction and ADT Specifications mechanisms: Procedures, function, and iterations as abstraction mechanisms; Parameterization mechanisms (reference vs. value). Type parameters and parameterized types, Modularization in programming languages; Object oriented language paradigm; Functional and logic language paradigms. Engineering of Programming Languages.

Course Content Sequencing

Weeks	**Detailed Course Outline**	**Allocated Time**
Week1,2	Overview of Programming Paradigm: Procedural, Object-oriented, Functional. Classification of Programming Paradigm Introduction Object Oriented Programming (OOP), Real-World Applications of C++ and Java, Java Script and Python programming. Introduction to Functional Programming, Logical Programming and its applications	**6 Hours**
Week2,3,4	Data Type and Data Structures: Scope(Global/ Local Scope), Binding and Visibility Relevance of data type with Java, Java Script, C++ and Python: Relevance of data structure with Java, Java Script, C++, Python • Linear data structure • Non – linear structure Relevance of Array with Java, Java Script, C++, Python **Grouping the class for Project presentations**	**9 Hours**

Week5,6	Language Description: Syntactic Structure (expression notations, abstract syntax tree, lexical syntax, grammar for expressions, variant of grammar)	**6 Hours**
Week7,8,9	Operational Semantics, Axiomatic Semantics AND Denotation. Interpreting Languages and RUN- TIME Considerations **Continuous Assessment I**: Group Project Presentations	**9 hours**
Week10,11,12	Presentations continues Parameterization Mechanism: Types of parameters and parameterized types, Reference VS Values, Activation record and storage management Tutorials **Continuous Assessment II: Written Test**	**9 Hours**
After Week 12	Examinations	

Recommended Reading Material

- Robert, W. S. (2012). *Concepts of Programming Languages* (10th ed.). New Jersey, USA: Pearson Education, Inc.

https://www.tutorialspoint.com/scope-and-lifetime-of-variables-in-java
https://www.tutorialspoint.com/computer_programming/computer_programming_data_types.htm

CSC405 - Operating System I (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Operating System I
Year of Study	IV
Course Code	CSC405
Credit Hours	3
Contact Hours	45
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment Final Examination	40% 60%
Total	**100%**

Course Lecturers and Instructor(s)	**Dr (Mrs) Taiwo Kolajo** **Mrs. Linda Okpanachi - Laboratory Instructor**

Course Description	This course will be covered in two semesters. In the first semester, this course examines the important problems in operating system design and implementation. The operating system provides an established, convenient, and efficient interface between user programs and the bare hardware of the computer on which they run. The operating system is responsible for sharing resources (e.g., disks, networks, and processors), providing common services needed by many different programs (e.g., file service, the ability to start or stop processes, and access to the printer), and protecting individual programs from interfering with one another. This course will introduce you to modern operating systems. We will focus on Windows-based operating systems, though we will also learn about alternative operating systems, including Linux. The course will begin with an overview of the structure of modern operating systems. Over the course of the subsequent units, we will discuss the history of modern computers, analyze in detail each of the major components of an operating system (from processes to threads), We will also explore process management.
Course Objectives	This course would enable the understanding of the following: 1. Train the students to understand the basic components of operating systems. 2. Enable students to understand interactions among the various components 3. Explore the issues involved in the design and development of operating systems
Learning Outcomes	At the end of the course, students will be able to: 1. Describe the general organization of a computer system. 2. Describe the components in a modern multiprocessor computer system. 3. Illustrate the transition from user mode to kernel mode. 4. Discuss how operating systems are used in various computing environments. 5. Provide examples of free and open-source operating systems. 6. Identify services provided by an operating system. 7. Illustrate how system calls are used to provide operating system services. 8. Compare and contrast monolithic, layered, microkernel, modular, and hybrid strategies for designing operating systems. 9. Illustrate the process for booting an operating system. 10. Apply tools for monitoring operating system performance. 11. Design and implement kernel modules for interacting with a Windows kernel. 12. Identify the separate components of a process and illustrate how they are represented and scheduled in an operating system. 13. Describe how processes are created and terminated in an operating system, including developing programs using the appropriate system calls that perform these operations. 14. Describe and contrast interprocess communication using shared memory and message passing. 15. Identify the basic components of a thread, and contrast threads and processes. 16. Describe the major benefits and significant challenges of designing multithreaded processes. 17. Illustrate different approaches to implicit threading, including thread pools, fork-join, and Grand Central Dispatch.

	18. Describe how the Windows and Linux operating systems represent threads. 19. Describe various CPU scheduling algorithms. 20. Assess CPU scheduling algorithms based on scheduling criteria. 21. Explain the issues related to multiprocessor and multicore scheduling. 22. Describe various real-time scheduling algorithms.	
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures, Recitations, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving and software modelling using Java or C Programming Languages	
Detailed Course Content	Definition of an operating system, the role of operating system in the overall computer system, Computer system organisation, computer system architecture, operating system operations, kernel data structures. Operating system services, user and operating system interface, system calls, system services, operating system design and implementation, operating system design and implementation. Process concept, process scheduling, operation on process, inter-process communication. Threads and concurrency, multithreading models. CPU scheduling concept, scheduling criteria, scheduling algorithms.	
Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**

Week1, 2	8. **Introduction to Operating Systems**
	• What operating systems do
	  • User view, system view, defining operating system
	• Computer organisation
	  • Interrupts, storage structure, I/O structure
	• Computer system architecture
	  • Single-Processor Systems, Multiprocessor Systems, Clustered Systems.
	• Operating-System Operations
	  • Multiprogramming and Multitasking, Dual-Mode and Multimode Operation, Timer.
	• Kernel Data Structures
	  • Lists, Stacks, and Queues, Trees, Hash function, Bitmaps
		6 Hours

Week3,4	9. **Operating System Structures**
	• Operating system services
	• User and Operating-System Interface
	  • Command Interpreters, Graphical User Interface, Touch-Screen Interface, Choice of Interface
	• System Calls
	  • Application Programming Interface, Types of System Calls
	• System Services
	• Linkers and Loaders
	• Operating-System Design and Implementation
	• Operating-System Structure
	  • Monolithic Structure, Layered Approach, Microkernels, Modules, Hybrid Systems
		9 Hours

Week5,6,7	10. **Process Management**
	• Process Concept
	  • The Process, Process State, Process Control Block, Threads
	• Process Scheduling
	  • Scheduling Queues, CPU Scheduling, Context Switch
	• Operations on Processes
	  • Process Creation, Process Termination
	• Interprocess Communication
	• IPC in Shared-Memory Systems
	• IPC in Message-Passing Systems
	  • Naming, Synchronization, Buffering
	11. **Continuous Assessment I**
		12 Hours

Week8	12. **Threads and Concurrency**
	• Overview
	  • Motivation, Benefits
	• Multicore Programming
	  • Programming Challenges, Types of Parallelism
	• Multithreading Models
	  • Many-to-One Model, One-to-One Model, Many-to-Many Model
		3 hours

Week9,10,11,12	13. **CPU Scheduling**
	• Basic Concepts
	  • CPU–I/O Burst Cycle, CPU Scheduler, Preemptive and Nonpreemptive Scheduling, Dispatcher
	• Scheduling Criteria
	• Scheduling Algorithms
	  • First-Come, First-Served Scheduling, Shortest-Job-First Scheduling, Round-Robin Scheduling, Priority Scheduling, Multilevel Queue Scheduling, Multilevel Feedback Queue Scheduling
	• Multi-Processor Scheduling
	  • Approaches to Multiple-Processor Scheduling, Multicore Processors, Load Balancing, Heterogeneous Multiprocessing
	• Real-Time CPU Scheduling
	  • Minimizing Latency, Priority-Based Scheduling, Rate-Monotonic Scheduling, Earliest-Deadline-First Scheduling, Proportional Share Scheduling
	14. **Continuous Assessment II**
		15 Hours

After Week12	15. Examination	

Recommended Reading Material

1. Wiiliam Stallings. (2018). Operating Systems: Internals and Design Principles, 9th Edition. ISBN 978-0-13-467095-9.
2. Michael Dahlin & Thomas Anderson. (2014). Operating Systems: Principles and Practice, 2nd Edition. ISBN-13: 978-0985673529, ISBN-10: 0985673524.
3. Andrew Tanenbaum & Herbert Bos. (2014). Modern Operating Systems, 4th Edition. ISBN 0-13-359162-X.

CSC407 - Machine Learning/Data Science (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE		
Faculty	Sciences	
Department	Computer Science	
Course Title	**Introduction to Machine Learning**	
Year of Study	IV	
Course Code	CSC407	
Credit Hours	3	
Contact Hours	36	
Pre-requisite(s)	CSC307	
Mode of Delivery	Classroom Lectures Term Paper Presentations (Group Work) Laboratory Practical Sessions	
Mode of Assessment		**Weight %**

Continuous Assessment Final Examination	40% 60%
Total	**100%**

Course Lecturers and Instructor(s)	**Dr. Emeka Emmanuel Ogbuju** **-Laboratory Instructors**
Course Description	This course is designed to introduce the students to the contemporary topics in Machine Learning and Data Sciences. Emphasis would be placed on industry-applications over theoretical knowledge.
Course Objectives	At the end of the course, the student should be able to: 1. understand some Machine Learning concepts (supervised and unsupervised). 2. understand and be able to implement some of the Machine Learning algorithms with python. 3. find statistical patterns, dimensionality reductions, clustering, classifications and predictions. 4. use any of the machine learning algorithms to solve business problems
Learning Outcomes	**At the end of the course, students will be able to:** 1. Design Machine Learning frameworks and methodologies for solving specific problems 2. Build machine learning models and evaluate them for efficient performance 3. Undertake projects that would solve problems in different domains using general data science skills.
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures, Group project Presentation, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving by major class groups using R/Rstudio Programming, Python Programming
Detailed Course Content	Introduction to Machine Learning, Supervised and Unsupervised algorithms, Discriminative Algorithms, Generative learning algorithms. Gaussian discriminant analysis. Naive Bayes, Bias/variance tradeoff and error analysis, Learning Theory, Regularization and Model Selection, Online Learning and the Perceptron Algorithm. (optional reading), Advice on applying machine learning, Linear/Logistic regressions, Dimensionality reductions, clustering, classification and predictions, Inference and learning with huge datasets, Introduction to Neural networks, Decision trees, Support Vector Machine(SVM), Introduction to data visualization and analytic, Evaluating and debugging learning algorithms, Introduction to Deep Learning Practical advice on structuring an ML projects. Introduction to Deep Learning.

Course Content Sequencing

Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	Introduction to Machine Learning, Supervised and Unsupervised algorithms,	**3 Hours**
Week2,3,4	Discriminative Algorithms, Generative learning algorithms. Gaussian discriminant analysis. Naive Bayes, Bias/variance tradeoff and error analysis, Learning Theory, Regularization and Model Selection, Online Learning and the Perceptron Algorithm. (optional reading),	**9 Hours**

Week,5,6	Advice on applying machine learning, Linear/Logistic regressions, Dimensionality reductions, clustering, classification and predictions, Inference and learning with huge datasets, Introduction to Neural networks, Decision trees, Support Vector Machine(SVM),	**6 Hours**
Week7,8	Introduction to data visualization and analytic, Evaluating and debugging learning algorithms, Introduction to Deep Learning	**6 hours**
Week9,10,11,12	Practical advice on structuring an ML project. Introduction to Deep Learning. **Project Implementations and Presentations** **Continuous Assessment II: Written Test**	**12 Hours**
After Week 12	Examinations	

Recommended Reading Material

1. Ian Goodfellow and Yoshua Bengio and Aaron Courville Deep Learning 2018, Wydawnictwo Naukowe PWN SA, 2018, 13: 978-0262035613, 10: 0262035618
2. Christopher M. Bishop. Pattern Recognition and Machine Learning 2019 Springer, 2006 13: 978-0387310732
3. Max Kuhn and Kjell Johnson. Applied Predictive Modelling 2016 Springer 13: 978-1461468486
4. Tom M. Mitchell. Machine Learning 2018 McGraw - Hill, 1997 13: 978-0070428072
5. Sebastian Raschka and Vahid Mirjalili. Python Machine Learning 2018 Packt Publishing, 2017 978-1783555130
6. Aurelien Geron Hands-On Machine Learning with Scikit -Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems 2019 New Delhi, SPD, 2017 good 978-1491962299

CSC409 - Net-Centric Computing (3 Units)

FEDERAL UNIVERSITY, LOKOJA COURSE OUTLINE		
Faculty	Sciences	
Department	Computer Science	
Course Title	Net-Centric Computing	
Year of Study	4	
Course Code	CSC409	
Credits Hours	3	
Contact Hours	42	
Prerequisite		
Mode of delivery	Lectures, Discussions and Practical	
Pre-requisite		
Mode of Assessment		Weight
Assignment/Discussions		20%
Continuous Assessment		20%
Final Examination		60%

Total		100%
Course Lecturers and Instructors	Dr Frederick Duniya BASAKY	
Course Description	Net-Centric Computing	
Course Objectives		
Learning Outcomes		
Teaching and Learning	The class will meet for three hours per week. The contact will be for lectures, discussions, presentation and defence of project work	
Detailed Course Content	.Background and History of Networking and Internet Network Architecture -Client/Server -Peer-to-Peer] The OS1-7 layer Reference model in general Network Protocols Physical and Data link layer Concepts (framing , error control ,flow control ,protocols) Internetworking and routing (Routing algorithms , internetworking ,congestion control) Transport Layer Services (connection establishment performance issues , flow and error control) Overview of Distributed Computing, -Overview of mobile and wireless computing -Fundamentals of Cryptography Authentication protocols Public-key algorithm –Types of attack e .g =denial of service =flooding =sniffing and =traffic redirection Basic Network tools & strategies -Intrusion detection -Fire wall -Detection of malware Kerberos .IPsec .Virtual Private Networks .Network Address Translation .Web technologies .Basic Server-side programs (php ,mySqL) .Basic Client Scripts (XHTML , XML , JavaScripts , CSS) .Nature of Client – server relationship .Web protocols with particular emphasis on HTTP . Support tools for website creation and web management . Building web application	
Weeks		
Week 1	Course introduction and outline presented to students	
Week 2 , 3 and 4	-Introductions to basics and concepts of the course, - Background and History of Networking and Internet -Network Architecture -Client/Server -Peer-to-Peer] - Introducing the Reference models	
Week 5 and 6	- **The OSI 7 Layers Reference models** - **Network Protocols**	

	- **Physical and Data link Layers and their concepts**
Week 7 and 8	- **Physical and Data link layer Concepts** (framing , error control ,flow control ,protocols) - Internetworking and routing (Routing algorithms , internetworking ,congestion control) - **Transport Layer Services** (connection establishment performance issues , flow and error control)
Week 9, 10 , and 11	- Overview of Distributed Computing, - Overview of mobile and wireless computing - Fundamentals of Cryptography Authentication protocols Public-key algorithm –Types of attack e .g =denial of service =flooding =sniffing and =traffic redirection
Week 12, 13 and 14	Basic Network tools & strategies -Intrusion detection -Fire wall -Detection of malware Kerberos .IPsec .Virtual Private Networks .Network Address Translation .Web technologies .Basic Server-side programs (php ,mySqL) .Basic Client Scripts (XHTML , XML , JavaScripts , CSS) .Nature of Client – server relationship .Web protocols with particular emphasis on HTTP . Support tools for website creation and web management . Building web application

CSC411 - Introduction to Cryptography and Network Security (2 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Introduction to Cryptography and Network Security
Year of Study	IV
Course Code	CSC411
Credit Hours	2
Contact Hours	24
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Tutorial Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Prof. Francisca O. Oladipo** **Mr Abdulwahab A. Jatto- Tutorial Sessions** **Mrs Linda O. Okpanachi-Tutorial Sessions**
Course Description	In this course the students will be equipped with the basic concepts of classical computer and network security paradigms. Also they will learn both theory and the applications for providing effective security in DBMS and intrusion detection.
Course Objectives	This course would enable the understanding of the following: 1. Provide students with knowledge of cryptographic systems as well as various mathematical background of importance to cryptography 2. Teach the students various techniques for identifying vulnerable target systems and attack patterns 3. Practically implement simple attacks to be tested on messages, or implement simple cryptosystem, or implement prime number generator 4. Explore the issues relating to security and different algorithms for key generation and cryptographic schemes
Learning Outcomes	At the end of the course, students will be able to: 1. Demonstrate an understanding of the fundamentals of Cryptography 2. Understand the standard algorithms used to provide confidentiality, integrity and authenticity; and define specifications for new and hybrid algorithms 3. Understand the various key distribution and management schemes. 4. Identify vulnerable target systems and attack patterns 5. Understand how to deploy encryption techniques to secure data in transit across data networks and build foundations to assess contemporary security policies and security mechanisms within organizations and illustrate the balance of the managerial and technical aspects of network security. 6. Discuss crypto-analysis methods used by intruders to break secure cryptosystems and current protocols for exchanging secured data 7. Implement Modular Arithmetic, perform system recovery 8. Develop symmetric and asymmetric cryptosystems 9. Perform an information systems audit using appropriate tools

Teaching and Learning	The class will meet for two hours each week. Class time will be used for a combination of Lectures, Recitations, while the Tutorial sessions would be organized 1 hour per week to focus on analyzing and solving applied numerical problems, using Python or Java for exploration of concepts under the supervision of teaching staff and participation of fellow students
Detailed Course Content	Basic definition of terms used in the text, Modular Arithmetic, Introduction/overview (includes concepts and levels of security), Public key cryptosystem, Diffie-Hellman, RSA, Security planning, Security policies. Identifying attack patterns, Analysis of security threats and risks. Design issues in security systems. Techniques to preserve confidentiality and authenticity against active attacks, Signature/certificate schemes, Secret sharing schemes, Multilevel secure DB for both Relational and OO relational DBs, Operational tools necessary for analysis and resolution of problems w.r.t effective filters and firewalls, tracing sources of attack and systems recovery, Security controls: System control, Input control, Processing control, Output control, Physical control of information assets, Discussion will be focused on physical protection of information assets from unauthorized access, Implementation and monitoring of security systems. Classical Encryption Techniques: Symmetric Cipher Model, Cryptography, Cryptanalysis and Brute-Force Attack, Substitution Techniques, Caesar Cipher, Monoalphabetic Cipher, Playfair Cipher, Hill Cipher, Polyalphabetic Cipher, One Time Pad. Block Ciphers and the data encryption standard: Traditional block Cipher structure, stream Ciphers and block Ciphers, Motivation for the feistel Cipher structure, the feistel Cipher, The data encryption standard, DES encryption, DES decryption, ADES example, results, the avalanche effect, the strength of DES. Cryptographic Hash Functions: Description, construction and properties, keyed vs. unkeyed, attacks, Iterated hash functions. Wireless network security: Wireless security, Wireless network threats, Wireless network measures, mobile device security, security threats, mobile device security strategy, IEEE 802.11 Wireless LAN overview, the Wi-Fi alliance, IEEE 802 protocol architecture. Security, IEEE 802.11i services, IEEE 802.11i phases of operation, discovery phase, Authentication phase, key management phase, and protected data transfer phase. Electronic Mail Security: Pretty good privacy, notation, operational; description, S/MIME, RFC5322, Multipurpose internet mail extensions, S/MIME functionality, S/MIME messages, S/MIME certificate processing, enhanced security services, Domain keys identified mail, internet mail architecture, EMail threats. Data Integrity, Authentication, MAC: Definition of a secure MAC

	algorithm, Applications of MAC algorithms (data integrity, data origin authentication), Generic attacks on MAC algorithms.	
Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**
Week1,2	**1. Cryptographic terms** • Basic definition of terms used in the text, Modular Arithmetic • Introduction/overview of cryptography • gcd, a|b, euclidean algorithm, extended euclidean algorithm • Public key cryptosystem, Diffie-Hellman, RSA etc. • Security of the RSA key generation process	**4 Hours**
Week3,4,5	**2. Security Planning, Policies and Control** • Security planning, Security policies. • Design issues in security systems. Techniques to preserve confidentiality and authenticity against active attacks. • Security controls • Implementation and monitoring of security systems. **Continuous Assessment I**	**9 Hours**
Week 6,7	**3. Classical Encryption Techniques** • Symmetric vs. Asymmetric/Public Key Cryptography • Block Ciphers • Data Encryption Standard (DES) • Block Cipher Modes • Other Symmetric Ciphers • Advanced Encryption Standard (AES) • One Time Pad	**6 Hours**
Week8,9	**4. Cryptographic Hash Functions** • Description, construction and properties	**7 hours**

	• keyed vs. unkeyed • attacks • Iterated hash functions • advanced attacks on hash functions 5. **Wireless Security and Cryptographic Protocols** • IEEE 802 protocol architecture. • Phases of operation, discovery phase, Authentication phase, key management phase, and protected data transfer phase. • Electronic Mail Security	
Week10,11,12	6. **Data Integrity, Authentication, MAC** • Definition of a secure MAC algorithm. • Applications of MAC algorithms (data integrity, data origin authentication). • Generic attacks on MAC algorithms • Confidentiality • Access control • Devastating attack by Fluhrer, Mantin and Shamir • Key recovery by a passive adversary **8. Class Discussion:** The crypto wars: should we have end-to-end encryption? **9. Mini Project:** Implementation of simple attacks to be tested on messages, or implementation of simple cryptosystem, or implementation of a prime number generator using Java or Python. **10. Continuous Assessment II**	**7 Hours**
After week 12	**11.** Examinations	

Recommended Reading Material

13. A.J. Menezes, P. van Oorschot and S.A. Vanstone. (1997). The Handbook of Applied Cryptography. CRC Press.
14. Nigel Smart. (2004) Cryptography: An Introduction (3rd Edition). McGraw-Hill College. ISBN-13 : 978-0077099879
15. Christof Paar, Jan Pelzl. (2010). Understanding Cryptography (1st Edition), Springer-Verlag Berlin Heidelberg. ISBN-978-3-642-44649
16. J. Katz and Y. Lindell. (2nd Edition) Introduction to Modern Cryptography (2nd edition), CRC Press, ISBN-13: 978-1466570269

CSC 421 – Information Technology Project Management (2 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE		
Faculty	Sciences	
Department	Computer Science	
Course Title	Introduction to Computer Science	
Year of Study	IV	
Course Code	CSC421	
Credit Hours	2	
Contact Hours	24	
Pre-requisite(s)	Nil	
Mode of Delivery	Classroom Lectures Case Studies Analysis	
Mode of Assessment		**Weight%**
Continuous Assessment Final Examination		40% 60%
Total		**100%**
Course Lecturers and Instructor(s)	**Prof. Francisca O. Oladipo**	
Course Description	Project Management is a very important area of information technology today as the primary challenge of managing IT projects is to develop an acceptable information system on schedule and within the allocated budget.	
Course Objectives	At the end of this course, students would: 1. Understand the growing need for better project management, especially for information technology projects. 2. Explain what a project is, provide examples of information technology projects, list various attributes of projects, and describe the triple constraint of projects. 3. Describe project management and discuss key elements of the project management framework, including project stakeholders, the project management knowledge areas, common tools and techniques, and project success factors. 4. Understand the role of the project manager by describing what project managers do, what skills they need, and what the career field is like for information technology project managers. 5. Understand Project Management Risks and Construct Risk Logs	

Learning Outcomes	At the end of the course, students will be able to: 14. Develop acceptable Project Management tools 15. Prepare a Project Management Plan 16. Manage mini projects based on both Prince2 and PMBoK Processes 17. Undertake Certifications on Project Management
Teaching and Learning	Lectures, Recitations, Tutorials: Key concepts would be taught during instructor-led sessions Laboratory: Project Management based on PRINCE2 and PMBoK
Detailed Course Content	Introduction: Definition, Phases and Processes, Methodologies, Importance and advantages, Standards: PRINCE2, PIMBOK. Contemporary Issues: PM Software and tools, Certification Programs and Courses, Human Resources and Staffing, IT Project Risk Management, IT Project Cost Management, Quality Assurance, Change Management.

Course Content Sequencing

Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	16. **Introduction and definition** • Definition of Project, Project Management and the underlying concept elements	**2 Hours**
Week2,3	17. **Phases and Processes** 18. Methodologies 19. Importance and advantages 20. **Continuous Assessment I**	**4 Hours**
Week 4,5, 6	21. Project Management Standards and certifications 22. PRINCE2, PIMBOK 23. Hands on and case studies using the two standards	**6 Hours**
Week 6,7,8	24. Contemporary Issues in Project Management 25. PM Software and tools 26. Certification Programs and Courses 27. Human Resources and Staffing	**6 hours**
Week 9,10	28. IT Project Risk Management 29. IT Project Cost Management	**4 hours**
Week 11,12	30. Quality Assurance 31. Change Management **Continuous Assessment II**	**4 hours**
After Week 12	32. Examinations	

Recommended Reading Material

17. The Project Management Institute (PMI) (2017): A Guide to the Project Management Body of Knowledge (6th edition).
18. Brett Harned (2017). Project Management for Humans: Helping People Get Things Done
19. Scott Berkun (2008). Making Things Happen: Mastering Project Management (revised edition)
20. Adam Josephs and Brad Rubenstein (2018). Risk Up Front: Managing Projects in a Complex World
21. Other resources

Introduction to project Management by Coursera https://www.edx.org/course/introduction-to-project-management

CSC 433 – Computer Graphics and Visualization (2 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Computer Graphics and Visualization
Year of Study	IV
Course Code	CSC433
Credit Hours	2
Contact Hours	24
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Mr. Ihinkalu Olalekan Ebenezer** **Mr. Isiaka Dauda/Mr. Paulinus Umeh (Practical Class)**
Course Description	The course covers general purpose graphics systems and their use. It gives an in depth knowledge of computer graphics and graphical user interfaces. This course introduce students to the concepts of graphical representation on computers and teach students the design of good graphical user interfaces.
Course Objectives	This course would enable the understanding of the following: a) Understand various Computer Visualization big ideas such as Image Representation, Scan Conversion, Color Modelling, n– Dimensional

	Transformation, primitive shape representation and rendering. b) Understand the underlying algorithms, mathematical concepts, supporting computer graphics such as composite 3D homogeneous matrices for translation, rotation, and scaling transformations, plane, surface normals, cross and dot products, hidden surface detection / removal, Scene graphs, display lists c) Show an understanding of how to communicate effectively using efficient visuals d) Discuss the application of computer graphics concepts in the development of computer games, information visualization, and business applications; as well as future trends in computer graphics and quickly learn future computer graphics concepts and APIs. e) Develop algorithms to identify and analyze misleading charts and visualizations using f) Specify the general software architecture of programs that use 3D computer graphics. g) Understand the hardware system architecture for computer graphics: graphics pipeline, frame buffers, and graphic accelerators/co-processors. h) Be able to use the 3D graphics API OpenGL. i) Select among models for lighting/shading: Color, ambient light; distant and light with sources; Phong reflection model; and shading (flat, smooth, Gourand, Phong); and current models for surfaces (e.g., geometric; polygonal; hierarchical; mesh; curves, splines, and NURBS; particle. j) Design and implement model and viewing transformations, the graphics pipeline and an interactive render loop with a 3D graphics API. k) Design and implement models of surfaces, lights, sounds, and textures (with texture transformations) using a 3D graphics API. l) Demonstrate familiarity with visual design by developing an interactive data visualization tool m) Present their skills in visual design and communication in a group project.
Learning Outcomes	At the end of the course, students will be able to: 1. Students should have an in depth knowledge of computer graphics and graphical user interfaces 2. Students will be able to use detailed example to solve problems 3. Students will able to understand the algorithms aspects of image synthesis 4. Also students should be able to write pseudocodes and explain what they are used for in computer graphics. 5. Students should be able to combine colors good colors for their front end in designing a software. 6. Student should Understand the hardware system architecture for computer graphics: graphics pipeline, frame buffers, and graphic accelerators/co-processors.

	7. And at the end of the course , they should be able to present their skills in visual design and communication in a group project.
Teaching and Learning	The class will meet for two hours each week. Class time will be used for a combination of Lectures, Seminar Presentation, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving and software modelling using C++, JAVA, and Python.
Detailed Course Content	Basic Definition of terms used in the text, Image Representation using: The RGB Color Model, Direct Coding, Lookup Table, Display Monitor, Printer, Image Files, Setting the Color Attribute of Pixels. Scan Conversion, Scan Converting primitive shapes (points, lines, rectangle, triangles etc.), Scan Converting a Circle, Scan Converting an Ellipse, Scan Converting Arcs and Sectors, Region Filling, Scan Converting a Character, Anti-Aliasing, Two–Dimensional, Transformations-Geometrics Transformations, Coordinate Transformation, Composite Transformation, Instance Transformation. **Image Representation**: The Digital Image, Raster Image Representation, Hardware Frame Buffers, Greyscale Frame Buffer, Pseudo-colour Frame Buffer, True-Colour Frame Buffer. **Colour Representation**: Additive vs. Subtractive Primaries, RGB and CMYK colour spaces, Greyscale Conversion, Hue, Saturation, Value (HSV) colour space. **Geometric Transformation** Rigid Body Transformations, Scaling, Shearing (Skewing) and Rotation. **Active vs. Passive Interpretation. Animation Hierarchies:** 3D Rigid Body Transformations, Rotation in 3D — Euler Angles, Rotation about an arbitrary axis in 3D. **Image Formation – 3D on a 2D display,** Perspective Projection and Orthographic Projection, Homography and its applications to Image, Digital Image Warping, Image-Based Rendering and Lighting, Graphics Pipeline and Rasterization, Ray Casting, tracing and Rendering, Graphics Hardware and Computer Games. Introduction to OpenGL Programming, Modelling and Matrices in OpenGL. Eigenmodel. Applications of Eigenvalue Decomposition in Computer Graphics. **Mathematical Background**: Points, Vectors and Notation. Basic Vector Algebra: Vector Addition, Vector Subtraction, Vector Scaling, Vector Magnitude, Vector Normalisation, Vector Multiplication, Dot Product and Cross Product. Reference Frames, Cartesian vs. Radial-Polar Form. **Matrix Algebra:** Addition, Scaling, Multiplication, Matrix Inverse and the Identity and Matrix Transposition. Basics of Computer Animation— Skinning/Enveloping, Particle Systems and ODEs, Hierarchical Modeling.

Course Content Sequencing

Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	• Introduction to Computer Graphics and Visualisation	**2 Hours**

Week2,3,	• Image Representation • Hardware Aspect, Plotter microfilm, Plotter Display, Graphic Tablets, Light Pens, and other Graphical input aids. Facsimile and its related problems. 1. Continuous Assessment I	**4 Hours**
Week4,5,6	• Scan Conversion (points, line, circle, ellipse…) • Refresh display refresh huggers, changing images, Light pen interaction • Two and Three Dimensional Transformation, perspective clipping algorithms. Hidden line removal bolded, surface removal.	**6 Hours**
Week7,8,9	• Warmock's method, shading, data reduction for graphical inputs. • Introduction to hand writing and character recognition. 2. Seminar Presentation On each Sub – Topics of Computer Graphics and Visualization, the students from the beginning of the semester will be shared in groups to present seminar topics and graded appropriately according to their group performance.	**6 Hours**
Week 10,11,12	• Curves synthesis and fitting • Contouring. Ring structures versus doubly linked lists • Hierarchical Structures • Data structures: Organization for intersotive graphics 3. Continuous Assessment II	**6 Hours**
After Week 12	4. Examinations	

Recommended Reading Material

1. Graphics & Visualization: Principles and Algorithms by T. Theoharis, G. Papaioannou, N. Platis & N.M Patrikalakis. http://graphics.cs.aueb.gr/cgvizbook/
2. Introduction to Computer Graphics by David J. Eck, Hobart & William Smith Colleges. https://open.umn.edu/opentextbooks/textbooks/420
3. Schaum's Outline of Computer Graphics 2nd Edition by Zhigang Xiang & Roy A. Plastock.
   https://www.amazon.com/Schaums-Outline-Computer-Graphics-Zhigang/dp/0071357815

4. Fundamentals of Computer Graphics- CM20219- Lecture Notes by Dr. John Collomosse, University of Bath, UK, http://personal.ee.surrey.ac.uk/Personal/J.Collomosse/pubs/cm20219.pdf

CSC 441 – Artificial Intelligence II (2 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Artificial Intelligence II
Year of Study	400
Course Code	CSC441
Credit Hours	2
Contact Hours	24
Pre-requisite(s)	CSC 208: Artificial Intelligence I
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	Dr. Edgar. O. Osaghae
Course Description	In this course teaches students different ways of approaching Artificial Intelligence (AI) problems.
Course Objectives	This course would enable the understanding of the following: 1. State the definition of Artificial Intelligence 2. List the different faculties involved with intelligent behavior 3. Explain the different ways of approaching AI 4. Look at some example systems that use AI 5. Describe the history of AI 6. Explain what an agent is and how it interacts with the environment 7. Identify the percepts available to the agent and the actions that the agent can execute, if given a problem situation 8. Measure the performance used to evaluate an agent 9. State based agents 10. Identify the characteristics of the environment 11. Describe the state space representation 12. Describe Some algorithms

	13. Formulate, when given a problem description, the terms of a state space search problem 14. Analyze the properties of Some algorithms 15. Analyze a given problem and identify the most suitable search strategy for the problem 16. Solve Some Simple problems 17. Explain Uninformed Search 18. List two types of Uninformed Search 19. Describe Depth First and Breadth First Search 20. Solve simple problems on Uninformed Search 21. Explain informed Search 22. Mention other names of informed Search 23. Describe Best-first Search 24. Describe Greedy Search 25. Solve simple problems on informed Search 26. Describe a Game tree 27. Describe Some Two-Player Games Search Algorithms 28. Explain Intelligent Backtracking 29. Solve Some Simple problems on tree search 30. Explain the meaning of Knowledge Representation (KR) 31. Describe the history of History of knowledge representation and reasoning 32. List some Characteristics of KR 33. List 4 main features of KR language 34. Describe the History of IPL 35. Discuss the similarities between Lisp and Prolog Programming 36. list the areas where Lisp can be used 37. Describe the history of natural language processing 38. List major tasks in NLP 39. Mention different types of evaluation of NPL 40. Explain an Expert System 41. Distinction between expert systems and traditional problem solving programs 42. Explain the term "Knowledge Base" 43. Explain the word Robotics 44. List 4 types of Robotics you know 45. Describe the history of Robotics

Learning Outcomes	At the end of the course, students will be able to: 1. Understand intermediate level of AI and then moves on to more advanced concepts. 2. Understand the description of search in artificial Intelligence - State Space Search, Uninformed Search, informed Search Strategies and Tree Search are also treated. 3. Understanding of Knowledge Representation and programming languages for AI. 4. Introduction to Artificial Intelligence and its applications – Expert System and Robotics.	
Teaching and Learning	The class will meet for two hours each week. Class time will be used for a combination of Lectures, Recitations, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving and software modelling using IPL programming, Lisp Programming Language and Prolog Programming Language.	
Detailed Course Content	Introduction to AI; What is Artificial Intelligent (AI)?, Introduction to Intelligent Agent (IA). Search in Artificial Intelligence; Introduction to State Space Search, Uninformed Search, Informed Search Strategies, Tree Search. Artificial Intelligence Techniques in Programming and Natural Languages, Knowledge Representation, Programming Languages for Artificial Intelligence, Natural Language Processing. Artificial Intelligence and its Applications; Expert System, Robotics.	
Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**
Week 1	**1. INTRODUCTION TO AI** i) What Is Artificial Intelligent (AI)? ii) Typical AI problem iii) Practical Impact of AI iv) Approaches to AI v) Limits of AI Today vi) AI History	**2 Hours**

Week 2, 3	**INTRODUCTION TO INTELLIGENT AGENTS** i) Introduction to Agent; Agent Performance, Examples of Agents, Agent Faculties, Intelligent Agents, Rationality, Bound Rationality. ii) Agent Environment; Observability, Determinism, Episodicity, Dynamism, Continuity, Presence of other Agents. iii) Agent Architectures or Reflex Agent; Table Based, Agent, Percept based, Subsumption Architecture, State- based Reflex Agent. **Continuous Assessment I**	**4 Hours**
Week 4	**INTRODUCTION TO STATE SPACE SEARCH** i) State space search; Goal Directed Agent, State Space, Search Notations. ii) Problem Space; Search Problem. iii) Examples of Search Problems; Illustration of a search process, Example problem: Pegs and Disks problem, Queens Problem, Problem Definition - Example, 8 puzzle. iv) Types of AI Search Techniques.	**2 Hours**
Week 5	**4. UNINFORMED SEARCH OR BRUTE FORCE SEARCH AND INFORMED SEARCH OR HEURISTIC SEARCH** i) Uninformed Search ii) Depth First and Breadth First Search; Depth First Search, Breadth First Search. iii) What is Heuristic?; Examples of Heuristic Function. iv) Best-first Search; Greedy Search, A* Search, Proof of Admissibility of A*, Proof of Completeness of A*, Properties of Heuristics, Using Multiple Heuristics. v) Beam search; Name and Uses, Extensions. vi) Hill climbing; Mathematical description, Variants.	**2 hours**

Week 6	**TREE SEARCH** Game Tree Two-Player Games Search Algorithms; Minimax Search, Alpha-Beta Pruning, Quiecence, Transposition Tables, Limited Discrepancy Search, Intelligent Backtracking	**2 Hours**
Week 7	**KNOWLEDGE REPRESENTATION** Overview of Knowledge Representation; Characteristics, History of Knowledge Representation and Reasoning. Knowledge Representation Languages.) Domain Modeling) Ontological Analysis Classic; The Classic Language, Enhancements to Classic.	**2 Hours**
Week 8	**7. PROGRAMMING LANGUAGES FOR ARTIFICIAL INTELLIGENCE (AI)** i) IA Programming Language; A taste of IA Programming Language, History of IA Programming Language. ii) Lisp Programming Language; History, Connection to Artificial Intelligence, Areas of Application, Syntax and Semantics. iii) Prolog Programming Language; History of Prolog, Prolog Syntax and Semantics.	**2 Hours**
Week 9	**8. NATURAL LANGUAGE PROCESSING** i) History of Natural Language Processing (NLP) ii) NLP using Machine Learning iii) Major Tasks in NLP iv) Statistical Natural Language Processing v) Evaluating of Natural Language Processing; Objectives, Sort History of Evaluation in NLP, Different Types of Evaluation.	**2 Hours**

Week 10	**9 EXPERT SYSTEM** i) What is an Expert System?; Comparison to Problem-Solving Systems. ii) Knowledge Base; Types of Knowledge Base. iii) Inference Engine; Architecture, The Recognize-Act Cycle, Data-Driven Computation versus Procedural Control, Inference Rules, Chaining. iv) Certainty Factors v) Real-Time Adaption; Ability to make Relevant Inquiries. vi) Knowledge Engineering v) General Types of Problems Solved vi) Different Types of Expert Systems vii) Examples of Applications Expert Systems viii) Advantages Expert Systems ix) Disadvantages Expert Systems	**2 Hours**
Week 11, 12	**10 ROBOTICS** i) What is a Robot; Types of Robot, History of Robots. ii) Components of Robots; Power Source, Actuation. iii) Sensing; Touch, Vision. iv) Manipulation; Mechanical Grippers, Vacuum Grippers, General Purpose Effectors. v) Locomotion; Rolling Robots, Walking Applied to Robots, Other methods of Locomotion, Environmental Interaction and Navigation, Human-Robot Interaction, Control, Robotics Research, Employment. **Continuous Assessment II**	**4 Hours**
After Week 12	**1.** Examinations	

Recommended Reading Material

1. Bowling, M. and Veloso, M. (2002). Multiagent Learning Using a Variable Learning Rate Artificial Intelligence, 136(2): 215-250.
2. Russell, Stuart J.; Norvig, Peter (2010). Artificial Intelligence: A Modern Approach (3rd ed.), Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-604259-7, p. 437-439.
3. NATIONAL OPEN UNIVERSITY OF NIGERIA, COURSE GUIDE ON ARTIFICIAL INTELLIGENCE, Published By: National Open University of Nigeria First Printed 2012 ISBN: 978-058-826-4

4. Charu C. Aggarwal (2021). Artificial Intelligence: A Textbook, Springer Nature Switzerland AG, Gewerbestrasse 11, 6330 Cham, Switzerland.

19.8 400 Level Second Semester

CSC400 – Research Project (6 units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Computer Graphics and Visualization
Year of Study	IV
Course Code	CSC400
Credit Hours	
Contact Hours	
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Seminar 1 – Topic Approval	20%
Seminar 2 – Proposal Defence	20%
Seminar 3 – Progress Report	20%
Final Examination	40%
Total	**100%**

Course Lecturers and Instructor(s)	**All Staff**
Course Description	The course is to strengthen the students' hand on skills on integration of the different skills into a programming project. Students are expected to develop a running computer application bigger in size. More emphasis will be put on creativity, robustness, data validation, security and completeness.
Course Objectives/ Learning Outcomes	This course would enable the understanding of the following: 1. To provide an avenue for students to integrate different subject areas into a single application 2. To sharpen the students problem solving skills 3. To improve student's ability to read on his/her own as an avenue for solving a certain real life problem
Detailed Course Content	Students should embark on Project works leading to substantial contribution to the field of Computer Science under the supervision of a departmental teaching staff. Students are expected to understand the art of final year project report writing, use systems analysis and design tools to design their respective projects, use a high-level/other relevant programming languages and a suitable compilers/interpreters to implement solutions to solve existing real-

	life societal problem, write a comprehensive project report to describe the details of objectives accomplished, methodology adopted, results obtained from projects and how to convert such results into products. Possible recommendations for improvement of the projects will also be provided. Seminars will be conducted at the end of the course. Students will also be expected to complete and present their respective final year project log books for assessment.

CSC 402 – Data Communication and Networking (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE		
Faculty	Sciences	
Department	Computer Science	
Course Title	Data Communication and Networking	
Year of Study	4	
Course Code	CSC402	
Credit Hours	3	
Contact Hours	42	
Pre-requisite		
Mode of Delivery	Class lectures, Discussion and Practical	
Mode of Assessment		Weight
Assignments Test Final Examination **Total**		20% 20% 60% **100%**
Course Lecturer and Instructor	Dr Frederick Duniya BASAKY	
Course Description	The Course Data Communication and Networking gives the students the required knowledge about the three key words, Data, Communication and Networking, their concepts and applications and how they are necessary to a graduate of computer science. The most important thing is the application of the course practically in real-life.	
Course Objectives	At the end of the study student should be able to; 1. Define Data, and difference the two types of data then give clear diagrams of the two types data 2. define communication. state and draw well labelled diagrams of the three communication modes, give life example of each of the three communication/connection modes 3. Introduction of Fourier Analysis 4. define a network and give life examples, define networking, give examples network protocols, Distributed networks. 5. draw the seven layers of Open System Interconnection (OSI), mention the functions and applications of each of the OSI layers and understand how the layers works in connection to data flow, communication technologies 6. how a network is configured and installed? How Routers and switches are configured and installed, Understand the concept of internetworking, 7. Understand the application of the fundamentals of Cryptography in Data, Communications and Networking 8. Draw the Four Layers of TCP/IP Layers, Draw a clear differentiate between	

	the two reference models (OSI and TCP/IP) 9. understand clearly the technology of the GSM, To list and give the functions of the networking and equipment 10. Configure and install the basic networking equipment (Switches, Routers, Modem and Hubs) 11. Define and give examples of network topologies
Learning Outcome	- The students at the end of the lectures; - were able to understand the basic concepts and applications of data communication and networking
Teaching and learning	- the lectures holds for three hours in a week with Practical holding in between.
Detailed Outline	- Define Data, - difference the two types of data - give clear diagrams of the two types data - define communication - state and draw well labelled diagrams of the three communication modes - give life example of each of the three communication/connection modes - understand the concept and application of Fourier Analysis - define a network and give life examples - define networking - give examples network protocols - draw the seven layers of Open System Interconnection (OSI) - mention the functions and applications of each of the OSI layers - understand how the layers works in connection to data flow, communication technologies - how a network is configured and installed? - How Routers and switches are configured and installed - Understand the concept of internetworking - Understand the application of the fundamentals of Cryptography in Data, Communications and Networking - Draw the Four Layers of TCP/IP Layers - Draw a clear differentiate between the two reference models (OSI and TCP/IP) - Understand clearly the technology of the GSM - To list and give the functions of the networking and equipment - Configure and install the basic networking equipment (Switches, Routers, Modem and Hubs) Define and give examples of network top
Course Content Sequencing	
Week1	**Introduction of the course to the class and course contents.**
Weeks 2, and 3	Define Data, and difference the two types of data then give clear diagrams of the two types data
Weeks 4 and 5	define communication. state and draw well labelled diagrams of the three communication modes, give life example of each of the three communication/connection modes Introduction of Fourier Analysis
Weeks 6	define a network and give life examples, define networking, give examples network protocols, Distributed networks.
Week7, 8 and 9	. draw the seven layers of Open System Interconnection (OSI), mention the functions and applications of each of the OSI layers and understand how the layers works in connection to data flow, communication technologies

	how a network is configured and installed? How Routers and switches are configured and installed, Understand the concept of internetworking,
Week 10 and 11	Understand the application of the fundamentals of Cryptography in Data, Communications and Networking Draw the Four Layers of TCP/IP Layers, Draw a clear differentiate between the two reference models (OSI and TCP/IP)
Weeks 12 and 13	understand clearly the technology of the GSM, To list and give the functions of the networking and equipment Configure and install the basic networking equipment (Switches, Routers, Modem and Hubs)
Week14	Define and give examples of network topologies, Collection and of assignments and administration of test

CSC404 – Operating System II (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Operating System II
Year of Study	IV
Course Code	CSC404
Credit Hours	3
Contact Hours	45
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment Final Examination	40% 60%
Total	**100%**

Course Lecturers and Instructor(s)	**Dr Taiwo Kolajo** **Laboratory Instructors**
Course Description	The second part of the course will be covered in this semester, a continuation of what we did in the first semester. This course will begin with process synchronization and go through memory managements (main memory and virtual memory), security and protection to file system interface. We will also explore Windows and Linux Operating systems as case studies. Programming exercises will be given to explore the above concepts.
Course Objectives	This course would enable the understanding of the following: 1. Train the students to understand synchronization techniques to achieve better performance of operating system 2. Enable students to understand main memory and virtual memory concepts 3. Explore the security and protection issues involved in the design and development of operating systems

Learning Outcomes	At the end of the course, students will be able to: 1. Describe the critical-section problem and illustrate a race condition. 2. Demonstrate how mutex locks, semaphores, monitors, and condition variables can be used to solve the critical-section problem. 3. Illustrate how deadlock can occur when mutex locks are used. 4. Define the four necessary conditions that characterize deadlock. 5. Identify a deadlock situation in a resource allocation graph. 6. Evaluate the four different approaches for preventing deadlocks. 7. Apply the banker's algorithm for deadlock avoidance. 8. Apply the deadlock detection algorithm. 9. Evaluate approaches for recovering from deadlock. 10. Explain the difference between a logical and a physical address and the role of the memory management unit (MMU) in translating addresses. 11. Apply first-, best-, and worst-fit strategies for allocating memory contiguously. 12. Explain the distinction between internal and external fragmentation. 13. Translate logical to physical addresses in a paging system that includes a translation look-aside buffer (TLB). 14. Describe hierarchical paging, hashed paging, and inverted page tables. 15. Define virtual memory and describe its benefits. 16. Illustrate how pages are loaded into memory using demand paging. 17. Apply the FIFO, optimal, and LRU page-replacement algorithms. 18. Describe how Linux, Windows 10, and Solaris manage virtual memory. 19. Explain the function of file systems. 20. Describe the interfaces to file systems. 21. Discuss file-system design tradeoffs, including access methods, file sharing, file locking, and directory structures. 22. Explore file-system protection. 23. Discuss security threats and attacks. 24. Explain the fundamentals of encryption, authentication, and hashing. 25. Examine the uses of cryptography in computing. 26. Describe various countermeasures to security attacks. 27. Discuss the goals and principles of protection in a modern computer system. 28. Explain how protection domains, combined with an access matrix, are used to specify the resources a process may access. 29. Examine capability- and language-based protection systems. 30. Describe how protection mechanisms can mitigate system attacks. 31. Explore the principles underlying Windows 10's design and the specific components of the system. 32. Provide a detailed discussion of the Windows 10 file system. 33. Illustrate the networking protocols supported in Windows 10. 34. Describe the interface available in Windows 10 to system and application programmers. 35. Describe the important algorithms implemented with Windows 10. 36. Examine the Linux process and thread models and illustrate how Linux schedules threads and provides interprocess communication. 37. Look at memory management in Linux.

	38. Explore how Linux implements file systems and manages I/O devices.	
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures, Recitations, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving and software modelling using Python	
Detailed Course Content	Process synchronisation, synchronisation tools. Deadlock, deadlock characterisation, methods for handling deadlock, deadlock prevention, deadlock avoidance, deadlock detection, recovery from deadlock. Main memory, contiguous memory allocation, paging, swapping. Virtual memory, demand paging, page replacement, allocation of frames, thrashing. File concept, access method, directory structure. Security problem, program threats, system and network threats, cryptography as a security tool, user authentication, implementing security defences. Goals of protection, principles of protection, domain of protection, access matrix, implementation of access matrix, revocation of access rights, role-based access control, mandatory access control (MAC), Windows and Linux operating systems as case studies.	
Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**

Week1,2,3	33. **Process Synchronisation**
	Synchronisation Tools
	• The Critical-Section Problem
	• Peterson's Solution
	• Hardware Support for Synchronization
	  • Memory Barriers, Hardware Instructions, Atomic Variables
	• Mutex Locks
	• Semaphores
	  • Semaphore Usage, Semaphore Implementation
	• Monitors
	  • Monitor Usage, Implementing a Monitor Using Semaphores, Resuming Processes within a Monitor
	Deadlocks
	• System Model
	• Deadlock in Multithreaded Applications
	  • Livelock
	• Deadlock Characterization
	  • Necessary Conditions, Resource-Allocation Graph
	• Methods for Handling Deadlocks
	• Deadlock Prevention
	  • Mutual Exclusion, Hold and Wait, No Preemption, Circular Wait
	• Deadlock Avoidance
	  • Safe State, Resource-Allocation-Graph Algorithm, Banker's Algorithm
	• Deadlock Detection
	  • Single Instance of Each Resource Type, Several Instances of a Resource Type, Detection-Algorithm Usage
	• Recovery from Deadlock
	  • Process and Thread Termination, Resource Preemption
		12 Hours

Week4,5,6,7	34. **Memory Management** **Main Memory** • Basic Hardware, Address Binding, Logical Versus Physical Address Space, Dynamic Loading, Dynamic Linking and Shared Libraries • Contiguous Memory Allocation • Memory Protection, Memory Allocation, Fragmentation • Paging • Basic Method, Hardware Support, Protection, Shared Pages • Structure of the Page Table • Hierarchical Paging, Hashed Page Tables, Inverted Page Tables • Swapping • Standard Swapping, Swapping with Paging, Swapping on Mobile Systems **Virtual Memory** • Demand Paging • Basic Concepts, Free-Frame List, Performance of Demand Paging • Page Replacement • Basic Page Replacement, FIFO Page Replacement, Optimal Page Replacement, LRU Page Replacement, LRU-Approximation Page Replacement, Counting-Based Page Replacement, Page-Buffering Algorithms, Applications and Page Replacement • Allocation of Frames • Minimum Number of Frames, Allocation Algorithms, Global versus Local Allocation, Non-Uniform Memory Access • Thrashing • Cause of Thrashing, Working-Set Model, Page-Fault Frequency, Current Practice • Memory Compression 35. **Continuous Assessment I**	**15 Hours**

Week7,8,9	36. **File System**
	• File Concept
	  • File Attributes, File Operations, File Types, File Structure, Internal File Structure
	• Access Methods
	  • Sequential Access, Direct Access, Other Access Methods
	• Directory Structure
	Single-Level Directory, Two-Level Directory, Tree-Structured Directories, Acyclic-Graph Directories, General Graph Directory
	
	37. **Security and Protection**
	Security
	• The Security Problem
	• Program Threats
	  • Malware, Code Injection, Viruses and Worms
	• System and Network Threats
	  • Attacking Network Traffic, Denial of Service, Port Scanning
	• Cryptography as a Security Tool
	  • Encryption, Implementation of Cryptography
	• User Authentication
	  • Passwords, Password Vulnerabilities, Securing Passwords, One-Time Passwords, Biometrics
	• Implementing Security Defenses
	  • Security Policy, Vulnerability Assessment, Intrusion Prevention, Virus Protection, Auditing, Accounting, and Logging, Firewalling to Protect Systems and Networks, Other Solutions
	Protection
	• Goals of Protection
	• Principles of Protection
	• Protection Rings
	• Domain of Protection
	  • Domain Structure
	  • Access Matrix
	  • Implementation of the Access Matrix
	  • Global Table, Access Lists for Objects, Capability Lists for Domains, Lock–Key Mechanism, Comparison
	• Revocation of Access Rights
	• Role-Based Access Control
	• Mandatory Access Control (MAC)
		10 hours

Week10,11,12	38. Case Studies • Windows Operating System • Linux Operating System 39. **Continuous Assessment II**	**9 Hours**
After Week 12	40. Examinations	

Recommended Reading Material

1. Wiiliam Stallings. (2018). Operating Systems: Internals and Design Principles, 9th Edition. ISBN 978-0-13-467095-9.
2. Michael Dahlin & Thomas Anderson. (2014). Operating Systems: Principles and Practice, 2nd Edition. ISBN-13: 978-0985673529, ISBN-10: 0985673524.
3. Andrew Tanenbaum & Herbert Bos. (2014). Modern Operating Systems, 4th Edition. ISBN 0-13-359162-X.

CSC 406 – Special Topics in Computer Science (3 Uints)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Special Topics(Internet Programming)
Year of Study	IV
Course Code	CSC406
Credit Hours	3
Contact Hours	36
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Term Paper Presentations Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers and Instructor(s)	**Dr. Victoria I. Yemi-Peters -Laboratory Instructors**
Course Description	General introduction to Computer Science as a field of study, the convergence of IT and the Real world, the role of the industry in Computing training, General skills required to succeed in the computing field. Information Technology Project Management, Round-trip Software Engineering and Applications Mordernization, Introduction to Machine Learning and data Science –general skills required to be a Data Scientist, Machine Learning Tools, Big Data, Internet of Things (IoTs), Cloud/Fog Computing, Bring Your Own Device (BYOD), Responsible Research and User Experience, Virtual Reality, Version Control Systems (Git and atom), Latest Developers Platform: Kotlin, Jenkins, Julia. This Content covers the topic, Internet Programming.
Course Objectives	This course would enable the understanding of the following: 1. Students will extend their knowledge of HyperText Markup Language (HTML) to incorporate multimedia elements into web pages, create HTML

	documents that incorporate images, tables, lists, forms and other HTML elements, use Cascading Style Sheets to control web page layout and format HTML elements. 2. Students will extend their knowledge of Cascading Style Sheets (CSS) to apply responsive web design techniques for mobile devices. 3. An introduction to the server-side/ client-side programming techniques used to develop interactive web sites.(Using technologies such as PHP and MySQL, JAVA SCRIPT) 4. Students will learn to create web sites that interact with web servers, manage user sessions, and store and retrieve data from databases. 5. Students will learn how to write client-side scripts in the JavaScript/PHP/C++ programming language that add interactivity and dynamic behaviors to web pages. 6 State the technological trends which have led to IoT, describe the impact of IoT on society, Name the core hardware components most commonly used in IoT devices and describe the interaction between software and hardware in an IoT device 7. Have the knowledge of the application of Cloud Computing as relates to the internet operations
Learning Outcomes	At the end of the course, students will be able to: 18. Design a good and interactive web platform 19. The Groups are expected to present a working project(web platform) temporarily hosted for Presentation as a part of their continuous assessment 20. Have a knowledge of IoT and its real life applications 21. Know the practice of using a network of remove servers hosted on the internet(Cloud Computing)
Teaching and Learning	The class will meet for three hours each week. Class time will be used for a combination of Lectures,Group project Presentation , Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving by major class groups using CSS,HTML, Java Script and PHP
Detailed Course Content	Introduction and definition, Internet Programming, Scripting Languages, introduction to the server-side/ client-side programming techniques used to develop interactive web sites. Strength and Weakness of PHP,CSS,HTML,JAVA SCRIPT. Introduction to Cloud Computing, Internet of Things.

Course Content Sequencing

Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	Introduction and definition, Internet Programming	**3 Hours**

Week2,3,4	An introduction to the server-side /client-side programming techniques HTML,CSS,PHP and JAVA SCRIPT **Continuous Assessment I: Grouping of the class for Project and Presentation(Implementation of a working web platform)**	**9 Hours**
Week,5,6	Introduction to IoT and its application Implementation of Group Projects (Discussions on progress)	**6 Hours**
Week7,8	Introduction to Cloud Computing: practice of using a network of remove servers hosted on the internet. Real life world applications	**6 hours**
Week9,10,11,12	Laboratory Practicals Project Implementations and Presentations **Continuous Assessment II: Written Test**	**12 Hours**
After Week 12	Examinations	

Recommended Reading Material

1. Cederholm D.(2011) Bulletproof Web Design: Improving Flexibility and Protecting Against Worst-Case Scenarios with HTML5 and CSS3 (Voices That Matter) 3rd Edition New Riders Pub; 299 pages
2. Monteiro M.(2012) Design is a Job. The Holy Grail for Creative Professionals everywhere. **ISBN:** 978-1-937557-04-1
3. Newcomer E(2002).Understanding Web Services- XML, WSDL, SOAP and UDDI
4. Holzner S.(2006) Ajax For Dummies® Wiley Publishing, Inc.

CSC408 – Expert System Technology (2 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Expert Systems
Year of Study	IV
Course Code	CSC408
Credit Hours	2
Contact Hours	24
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions
Status	Elective

Mode of Assessment	**Weight%**
Continuous Assessment Final Examination	40% 60%
Total	**100%**

Course Lecturers and Instructor(s)	**Prof. Francisca O. Oladipo** **Mr. Paulinus Umeh -Laboratory Instructor**
Course Description	In this course the student will learn the methodology used to transfer the knowledge of a human expert into an intelligent program that can be used to solve problems.
Course Objectives	This course would enable the understanding of the following: 1. Provide students with the understanding of the technology of Expert Systems 2. Train the students in the process of developing an expert systemin a specific domain 3. Develop in the students, the abilities to apply, build and modify rule-based systems to solve real problems, 4. Explore the issues involved in the design and development of Expert Systems and the process of transfer of human knowledge to a machine.
Learning Outcomes	At the end of the course, students will be able to: 1. Define an Expert System and identify the components 2. Apply the methodology to transfer human knowledge into an expert system and develop a prototype Expert System in different problem domains 3. Apply the different domains of learning to implement a rule-based system, while applying knowledge representation and design a knowledge base 4. Evaluate Expert System tools and implement a rule-based expert system 5. Carry out Genetic Algorithm Project based on Decision Support System 6. Use various knowledge representation methods and different expert system structures from the industrial engineering point of view
Teaching and Learning	The class will meet for two hours each week. Class time will be used for a combination of Lectures, Recitations, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving and software modelling using Python

Detailed Course Content	Introduction and definition, Knowledge-Based Systems (KBS), Expert Systems (ES). Data/Information/Knowledge. Knowledge representations and mappings, approaches and issues (e.g. predicate logic, fuzzy logic, week and strong slot and filler structures), Semantic Nets, Rules, Frames, Scripts, Logic, RDF. Knowledge acquisition, the frame problem, symbolic reasoning under uncertainty (nonmonotonic reasoning, augmenting a problem Solution). Reasoning and Inference: Predicate Logic, Description Logics, Inference Methods, Resolution. Reasoning and Inference: Inference Methods, Resolution. ECLiPSe-specific Language features, Structure, Iteration, Loops, I/O), statistical reasoning (e.g. probability and Bays Theorem, Bayesian networks, Dumpster-Shafer theory), building knowledge-based systems. Reasoning with Uncertainty. Probability, Bayesian Decision Making, Dempster-Shafer Theory, Reasoning with Uncertainty, Dempster-Shafer Theory, Approximate Reasoning, Fuzzy Logic. Semantic Web Technologies, Semantic Web Technologies, KBS Case Studies	
Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	41. **Introduction and definition** • Definition of the underlying concept elements of Expert Systems • Knowledge-Based Systems (KBS), • Expert Systems (ES). • Expert Systems technology • Data/Information/Knowledge	**3 Hours**
Week2,3	42. **Knowledge representations and mappings**, 43. Approaches and issues in Knowledge representation (e.g. predicate logic, fuzzy logic, week and strong slot and filler structures), Semantic Nets, Rules, Frames, Scripts, Logic, RDF. Knowledge acquisition, the frame problem, symbolic reasoning under uncertainty (nonmonotonic reasoning, augmenting a problem Solution). 44. **Continuous Assessment I**	**4 Hours**

Week4,5,6	45. **Knowledge representations and mappings, approaches, and issues** 46. predicate logic 47. fuzzy logic 48. week and strong slot and filler structures 49. Semantic Nets, Rules, Frames, Scripts, Logic, RDF. 50. **Knowledge acquisition** 51. the frame problem 52. symbolic reasoning under uncertainty (nonmonotonic reasoning, augmenting a problem Solution)	**6 Hours**
Week 7,8,9	53. **Reasoning and Inference** 54. Predicate Logic, Description Logics, Inference Methods, Resolution. 55. Inference Methods, Resolution. 56. **Laboratory Session** 57. ECLiPSe-specific Language features, Structure, Iteration, Loops, I/O) 58. **statistical reasoning** (e.g. probability and Bays Theorem, Bayesian networks, Dumpster-Shafer theory), building knowledge-based systems. Reasoning with Uncertainty.	**6 hours of classes and lab**
Week 10,11	59. Probability, Bayesian Decision Making, Dempster-Shafer Theory, Reasoning with Uncertainty, Dempster-Shafer Theory, Approximate Reasoning, Fuzzy Logic. Semantic Web Technologies, Semantic Web Technologies,	**4 Hours**
Week 12	60. **Continuous Assessment II**: Team project, KBS Case Studies	**4 Hours**
After Week 12	61. Examinations	

Recommended Reading Material

22. Peter Nikolopoulos. (1997). Expert Systems: Introduction to First and Second Generation and Hybrid Knowledge Based Systems 1st Edition. ISBN-13: 978-0824799274
23. Peter J.F. Lucas & Linda C. van der Gaag. (1991). Principles of Expert Systems, Centre for Mathematics and Computer Science, Amsterdam, Addison-Wesley.
24. Bryan S. Todd, an introduction to Expert Systems, Technical Monograph, University of Oxford, UK

25. Cornelius Leondes. (ed 2001). Expert Systems. The Technology of Knowledge Management and Decision Making for the 21st Century. Academic Press, 1st Edition

CSC412 – Compiler Construction II (C++ and Java) (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Compiler Construction II
Year of Study	IV
Course Code	CSC412
Credit Hours	3
Contact Hours	36
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%
Total	**100%**

Course Lecturers	**Terungwa Simon Yange**
Course Description	This course introduces students to the concept of automata, grammars, LR table and complexity. It sets a background for more advanced computer studies like principles of programming languages.
Course Objectives	At the end of this course, the student should 1. Know types of parsers/parsing 2. Know how to use compiler construction tools, such as generators of scanners and parsers 3. Be familiar with assembly code and virtual machines, such as the JVM, and bytecode 4. Be able to define LL, LR, and LALR grammars **5.** Be familiar with compiler analysis and optimization techniques

Learning Outcomes	At the end of the course, students will be able to: 1. Understand all the basic and advanced concepts of Compiler Construction 2. 2. Students should also be able to develop a simple compiler using either C++ or Java or both programming languages 3. Students should be able to apply knowledge gained from compiler construction courses to solve real-life problem.
Teaching and Learning	1. **Lectures**: contents of the course will be presented and taught to students in the classroom. Classroom teachings will be supported with practical examples. 2. **Projects**: Group and individual projects will be given to students to solve practical problems in compiler construction. Students will be expected to come to the classroom individually and defend their respective individual projects. 3. **Assignments**: Students will be asked to solve class assignments with respect to topics covered in the class to examine, test the understanding of and reveal the state of assimilation of the course contents by students. 4. **Term Papers**: Students will be asked to write comprehensive term papers on selected sub-topics within the compiler construction course contents. The term papers will help students develop their understanding and in-depth analysis of components of the course contents. In some situations, students will be given a typical compiler construction research paper and will be asked to study, analyze and summarize in their own understanding, gaps and findings from the contents of the paper. Such term papers however will be subjected to thorough plagiarism checks.
Detailed Course Content	Extensive revisions of Compiler Construction I as a foundation and prerequisite for the course, Grammars and Languages - context free grammars -parts 1,2,3, Top-down and bottom-up language - Top-down parsing, bottom-up parsing, Run-time storage organization, The use of display in run-time storage allocation, LR grammars and analyzers, Construction of LR table-table-driven stack parser, Organization of Symbol table, Allocation of storage to run-time variables - Compiling to a register-oriented architecture, Code Generation, Optimization with systems- Optimization, Translator with

	systems- Recursive-descent translation, recursive-descent parsing, Introduction to Java CC, Introduction to Yacc, LL(1) Grammars, Practical's on designing and developing a Simple Compiler S1- using Java or C++ programming language.	
Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	7. **Recap of Compiler Construction I** • Introduction to translators (Compilers, Interpreters and Assemblers) • Explain the phases of a compiler • Describe the internal structure of a compiler • Describe grammars and languages • Explain regular expressions	**3 Hours**
Week2,3	8. **Parsing and Memory Organization** • Top-down and bottom-up language - Top-down parsing, bottom-up parsing, • Run-time storage organization • The use of display in run-time storage allocation • Allocation of storage to run-time variables - Compiling to a register-oriented architecture **Continuous Assessment I**	**6 Hours**
Week4,5,6	9. LL, LR, and LALR grammars and analyzers, Construction of LL, LR, and LALR table- table-driven stack parser, Organization of Symbol table,	**9 Hours**
Week7,8	10. Code Generation, Optimization with systems- Optimization, Translator with systems- Recursive-descent translation, recursive-descent parsing	**6 hours**
Week9,10,11,12	11. Introduction to Java CC, Introduction to YACC, Practical's on designing and developing a Simple Compiler S1- using Java or C++ programming language **Revision and Continuous Assessment II**	**12 Hours**
After Week 12	12. Examinations	
Recommended Reading Material 1. Aho, Alfred & Sethi, Ravi & Ullman, Jeffrey. *Compilers: Principles, Techniques, and Tools* ISBN 0201100886 The Classic Dragon book.		

2. Appel*, A., Modern Compiler Implementation in Java,* 2nd ed., Cambridge University Press, 2002.
3. Appel, Andrew *Modern Compiler Implementation in C/Java/ML* (respectively ISBN 0-521-58390-X,ISBN 0-521-58388-8, ISBN 0-521-58274-1) is a set of cleanly written texts on compiler design, studied from various different methodological perspectives.
4. Brown, P.J. *Writing Interactive Compilers and Interpreters* ISBN 047127609X Useful practical advice, not much theory.
5. Fischer, Charles & LeBlanc, Richard. *Crafting A Compiler* ISBN 0805332014 Uses an ADA like pseudo-code.
6. Fischer, LeBlanc, Cytron*, Crafting a Compiler Implementation*, Addison-Wesley
7. Holub, Allen *Compiler Design in C* ISBN 0131550454 Extensive examples in "C".
8. Hunter, R. *The Design and Construction of Compilers* ISBN 0471280542 several chapters on theory of syntax analysis, plus discussion of parsing difficulties caused by features of various source languages.
9. Keith, D. Cooper & Linda Torczon, "Engineering a Compiler", Morgan Kaufmann Publishers, 2004
10. Pemberton, S. & Daniels, M.C. *Pascal Implementation. The P4 Compiler* ISBN 0853123586 (Discussion) and ISBN 085312437X (Compiler listing) Complete listing and readable commentary for a Pascal compiler written in Pascal.
11. Randy Allen and Ken Kennedy, "Optimising Compilers for Modern Architectures", Morgan Kaufmann Publishers, 2001.
12. Weinberg, G.M. *The Psychology of Computer Programming: Silver Anniversary Edition* ISBN 0932633420 Interesting insights and anecdotes.
13. Wirth, Niklaus *Compiler Construction* ISBN 0201403536 From the inventor of Pascal, Modula-2 and Oberon-2, examples in Oberon.

CSC 422 – Computer System Performance Evaluation (2 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Computer Systems Performance Evaluations
Year of Study	IV
Course Code	CSC422
Credit Hours	2
Contact Hours	24
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures Laboratory Practical Sessions

Mode of Assessment	**Weight%**
Continuous Assessment Final Examination	40% 60%
Total	**100%**
Course Lecturers and Instructor(s)	Dr. Edgar. O. Osaghae
Course Description	In this course the students will learn how to measure the performance of a Computer System
Course Objectives	This course would enable the understanding of the following: 1. Teach the fundamental concepts behind the tools and techniques used in computer-systems performance analysis. 2. Know sound scientific methodologies to use when conducting performance evaluation experiments. 3. Comparing the performance of different computer systems and use the results for decision making. 4. To understand how to interpret performance evaluation results in a meaningful way
Learning Outcomes	At the end of the course, students will be able to: 1. Understand the concepts of performance evaluation techniques and their methodologies. 2. Know how to where to measure and how to measure different components of in a Computer System. 3. Use sound scientific methodologies in the right perspectives in order to avoid the risk of producing wrong analysis results. 4. Know the use of various simulation packages/tools.
Teaching and Learning	The class will meet for two hours each week. Class time will be used for a combination of Lectures, Recitations, Tutorials and Laboratory Practical Sessions. Key concepts would be taught during instructor-led sessions, while the Laboratory sessions will be based on problem-solving and software modelling using C++ programming.
Detailed Course Content	Introduction; Measuring Performance, Common Goals of Performance Analysis, Solution Techniques. Metrics of Performance; What is a Performance Metric? Characteristics of a Good Performance Metric, Processor and System performance Metrics, Other types of Performance Metrics, Speedup and Relative Change, Means versus end Metrics. Average Performance and Variability; Why Mean Values? Indices of Central Tendency, Other types of Means, Quantifying Variability. Errors in Experimental Measurements; Accuracy, Precision and Resolution, Sources of

	Errors, A Model of Errors, Quantifying Errors. Comparing Alternatives; Comparing two Alternatives, Comparing more than two Alternatives. Measurement Tools and Techniques; Events and Measurement Strategies, Interval Timers, Program Profiling, Event Tracing, Indirect and ad hoc Measurements, Perturbations due to measuring. Benchmark Programs; Types of Benchmark Programs, Benchmark Strategies, Example Benchmark Programs. Linear-Regression Models; Least-Squares Minimization, Confidence Intervals for Regression Parameters, Correlation, Multiple Linear Regression, Verifying Linearity, Nonlinear Models. The Design of Experiments; Types of Experiments, Terminology, Two-Factor Experiments, Generalized m-factor Experiments, $n2^m$ Experiments. Simulation and Random-Number Generation; Simulation-Efficiency Considerations, Types of Simulations, Random-Number Generation, Verification and Validation of Simulations.	
Course Content Sequencing		
Weeks	**Detailed Course Outline**	**Allocated Time**
Week 1	**1. Introduction** • Measuring performance • Common goals of performance analysis • Solution techniques	**2 Hours**
Week 2, 3	**2. Metrics of Performance** • What is a Performance Metrics? • Characteristics of a good performance metric • Processor and system performance metrics • Other types of performance metrics • Speedup and relative change • Means versus ends metrics **Continuous Assessment I**	**4 Hours**
Week 4	**3. Average Performance and Variability** • Why mean values? • Indices of central tendency • Other types of means • Quantifying variability	**2 Hours**
Week 5	**4. Errors in Experimental Measurements** • Accuracy, precision, and resolution • Sources of errors • A model of errors • Quantifying errors	**2 hours**
Week 6	**5. Comparing Alternatives** • Comparing two alternatives • Comparing more than two alternatives	**2 Hours**

Week 7	**6. Measurement Tools and Techniques** • Events and measurement strategies • Interval timers • Program profiling • Event tracing • Indirect and ad hoc measurements • Perturbations due to measuring	**2 Hours**
Week 8	**7. Benchmark programs** • Types of benchmark programs • Benchmark strategies • Example benchmark programs	**2 Hours**
Week 9	**8. Linear-Regression Models** • Least-squares minimization • Confidence intervals for regression parameters • Correlation • Multiple linear regression • Verifying linearity • Nonlinear models	**2 Hours**
Week 10	**9. The Design of Experiments** • Types of Experiments • Terminology • Two-Factor Experiments • Generalized m-factor Experiments • $n2^m$ Experiments	**2 Hours**
Week 11, 12	**10. Simulation and Random-Number Generation** • Simulation-efficiency Considerations • Types of Simulations • Random-Number Generation • Verification and validation of simulations **Continuous Assessment II**	**4 Hours**
After Week 12	**11.** Examinations	

Recommended Reading Material

1. David J. Lilja. (1997). Measuring Computer Performance: A Practitioner's Guide, Cambride University Press, Trumpington Street, Cambridge, United kingdom.
2. Kant K. (1992). Introduction to Computer System Performance Evaluation, McGraw-Hill Inc., United Kingdom.
3. Haverkort B. R. (1998). Performance of Computer Communication Systems, John Wiley and Sons Ltd.

4. Harchol-Balter M. (2013). Performance Modeling and Design of Computer Systems, Cambridge University Press, United Kingdom.

CSC424 – Modelling and Simulation (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE		
Faculty	Sciences	
Department	Computer Sciences	
Year of Study	4	
Course Code	CSC 424	
Credit Hours	3	
Contact Hours	42	
Pre-requisite		
Mode of Delivery	Lectures, Practicals	
Mode of Assessment	Assignment, Test, Project and Examination	
Presentation/Project		20%
Continuous Assessment		20%
Final Examination		60%
Total	100%	
Course Lecturers and Instructors	Dr Frederick Duniya BASAKY	
Course Description	Modelling and Simulation	
Course Objectives	At the end of the course students should be able to; - know the basic concepts of modelling and simulation - Know the applications of modelling and simulation, - Understand simulation processes - Solve real life problems applying modelling and simulation, - Model and simulate some basic statistical distribution theory	
Learning Outcome		
Teaching and Learning	This course, 3- credit units will meet three times for lectures. Discussion on the project	
Detailed Course Content	-Introduction to the Modelling and Simulation, - What is a Model? -What is Modelling? -What is Stimulation -Introductory definitions -Basic uses of Modelling and Simulation - Simulation processes -Some basic statistical distribution theory Queues - Basic components of queues - Queueing rules -Special types of Queues	

	-Stochastic process, - Application of Stochastic process - Discrete state and continuous state process and their applications - Poisson Process and - Applications of Poisson Process -Random numbers and applications
Course Content Sequencing	
Week1	Introducing the course and the course contents for the semester
Weeks 2 and 3	-Introduction to the Modelling and Simulation,
Weeks 4, 5 and 6	- What is a Model? What is Modelling? -What is Stimulation -Introductory definitions -Basic uses of Modelling and Simulation - Simulation processes
Weeks 7,and 8	Some basic statistical distribution theory Queues - Basic components of queues - Queueing rules -Special types of Queues
Weeks 9, 10 and 11	Stochastic process, - Application of Stochastic process - Discrete state and continuous state process and their applications - Poisson Process and - Applications of Poisson Process -Random numbers and applications
Weeks 12 and 13	Projects presentation and defence and assessment
Week14	Revision

CSC432 – Formal Methods in Software Engineering (3 Units)

FEDERAL UNIVERSITY LOKOJA COURSE OUTLINE	
Faculty	Sciences
Department	Computer Science
Course Title	Formal Methods in Software Engineering
Year of Study	IV
Course Code	CSC432
Credit Hours	3
Contact Hours	36
Pre-requisite(s)	Nil
Mode of Delivery	Classroom Lectures

Mode of Assessment	**Weight%**
Continuous Assessment	40%
Final Examination	60%

Total	**100%**
Course Lecturers	**Terungwa Simon Yange**
Course Description	Formal Methods supports the production of highly-reliable Software. In this course, students would learn a collection of techniques for formal software development, spanning the whole development process: from high-level semantic modeling to coding and debugging. The study will not be done in the abstract, however, but through the use of actual tools supporting these techniques.
Course Objectives	This course would enable the understanding of the following: 1. The precise specification of run-time properties that a software system is expected to satisfy through the construction of highly reliable Software 2. CSP and UML for Software Construction 3. How to solve problems with specifications that are precise in formal syntax, semantics, and theory 4. How to define unambiguous, high-quality specifications, and build provide a background for automated tool support. 5. How to construct highly automated verification tools that help software developers analyze specifications and corresponding code, looking for errors in requirements, models, designs, and implementations.
Learning Outcomes	At the end of the course, students will be able to: 1. Demonstrate an understanding of the precise specification of run-time properties that a software system is expected to satisfy through the construction of highly reliable Software 2. Deploy CSP and UML for Software Construction 3. Define and solve problems with specifications that are precise in formal syntax, semantics, and theory 4. Define unambiguous, high-quality specifications, and build provide a background for automated tool support. 5. Construct highly automated verification tools that help software developers analyze specifications and corresponding code, looking for errors in requirements, models, designs, and implementations.
Teaching and Learning	5. **Lectures:** Detailed content of course are taught in class using problem solving approaches 6. **Presentations:** Course contents are shared among students to research on. Students are grouped or assigned work to individually present. This is done for the purpose of self-reading improvement and student assessment. 7. **Laboratory Practical:** Laboratory sessions will be based on

	Problem solving and software modelling using Python, Alloy, Lustre, Dafny Static Analysis using Java 8. **Project:** Students will be required to formulate problems in the different areas of the course content, design the solutions to these problems and solve them.
Detailed Course Content	Introduction: Mathematical Foundations - Naive set theory, Propositional logic, First-order logic, Reasoning about programs: invariants, Hoare-Logic, Termination. Formal specifications: the Z notation, Reasoning about specifications, Refining specifications into (Python) code. Software Specification: High-level semantic design, System design and behavioural properties, Code-level properties. Main Software Validation Techniques Model Finding/Checking: often automatic, abstract Deductive Verification: typically semi-automatic, precise (source code level) Abstract Interpretation: automatic, correct, not complete, terminates. Design and model software systems in the Alloy language. Write system and property specifications in Lustre language, Recap of basic notions in set theory. Relations and relational operators. Modelling general software systems. Introduction to the Alloy modelling language (or any other modelling language of instructor's choice). Alloy's foundations. Signatures, fields and multiplicity constraints. Modelling simple domains in Alloy. Generating and analysing model instances with the Alloy Analyzer. Relations and operations on them. Formulas, Boolean operators and quantifiers. Expressing constraints on relations using Alloy formulas. Facts and assertions. Checking models and assertions with the Alloy Analyzer. Practice with writing Lustre models and expressing their properties. Checking properties via synchronous observers. Simulating Lustre programs with the Kind 2 tool. Specifying and verifying programs in high-level programming languages. Introduction to Dafny. Main features. Specifying pre and post-conditions.

Course Content Sequencing

Weeks	**Detailed Course Outline**	**Allocated Time**
Week1	**1. Mathematical Foundations and Formal specifications** • Naive set theory • Propositional logic and First-order logic • Reasoning about programs: invariants, Hoare-Logic, Termination. • Z notation, Reasoning about specifications, Refining specifications into (Python) code.	**3 Hours**

Week2,3	2. Software Specification: High-level semantic design, System design and behavioural properties, Code-level properties. Main Software Validation Techniques Model Finding/Checking: often automatic, abstract Deductive Verification: typically semi-automatic, precise (source code level) Abstract Interpretation: automatic, correct, not complete, terminates. **Continuous Assessment I**	**6 Hours**
Week4,5,6	3. Design and model software systems in the Alloy language. Write system and property specifications in Lustre language, Recap of basic notions in set theory. Relations and relational operators. Modelling general software systems. Introduction to the Alloy modelling language (or any other modelling language of instructor's choice). Alloy's foundations.	**9 Hours**
Week7,8	4. Signatures, fields and multiplicity constraints. Modelling simple domains in Alloy. Generating and analysing model instances with the Alloy Analyzer. Relations and operations on them. Formulas, Boolean operators and quantifiers. Expressing constraints on relations using Alloy formulas.	**6 hours**
Week9,10,11,12	5. Facts and assertions. Checking models and assertions with the Alloy Analyzer. Practice with writing Lustre models and expressing their properties. Checking properties via synchronous observers. Simulating Lustre programs with the Kind 2 tool. Specifying and verifying programs in high-level programming languages. Introduction to Dafny. Main features. Specifying pre and post-conditions. **Revision and Continuous Assessment II**	**12 Hours**
After Week 12	6. Examinations	

Recommended Reading Material

1. Heitmeyer, C. L., Jeffords, R. D., & Labaw, B. G. (1996). Automated Consistency Checking of Requirements Specifications. ACM Transactions on Software Engineering and Methodology, 5(3), 231-261.
2. van Vliet, H. (1999). "Software Engineering: Principles and Practice (2nd Edition)" Wiley.
3. Spivey, J. M. (1998). The Z Notation: A Reference Manual 2nd Edition
4. John C Martin Introduction to languages and the Theory of Computation.
5. Mishra K. L.P. and Chandrashekaran N. (2008). Theory of Computer Science – Automata languages and computation -, 3rd edition, Prentice-Hall, India