MAKERERE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATION SCIENCES

SCHOOL OF COMPUTING AND INFORMATICS TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

P.O. BOX 7062, KAMPALA, UGANDA

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

(BSC.CS) DEGREE PROGRAMME

REVISED FEBRUARY 2013

(DAY/ EVENING PROGRAMME)

TO RUN – AUGUST 2013/2014

Contents

Bachelor of Computer Science 6

1. Background to the B.Sc Computer Science Program 6

1.1 Who is a Computer Science Graduate? 6

2. Rationale for reviewing the Programme 6

3. The Program 7

3.1 Target Group 7

3.2 Nature of the Program 7

3.3 Duration 7

3.4 Tuition Fees 7

4. Regulations 7

4.1 Admission requirements 7

(a) Direct Entry 7

(b) Diploma holders: 8

4.2 Progression 8

4.2.1 Normal Progress 8

4.2.2 Probationary 8

4.2.3 Discontinuation 8

4.2.4 Retaking a Course 8

4.3 Weighting System 9

4.4 Semester Load and Minimum Graduation Load 9

4.5 Course Assessments 9

4.6 Grading and Pass mark 9

4.7 Minimum Pass Mark 9

4.8 Calculation of Cumulative Grade Point Average (CGPA) 9
4.9.1Content Distribution by Knowledge Area 10
5. The Curriculum 11
5.1 Course Outline 11
5.1.1 Semester 1 (5 Courses) 11
5.1.2 Semester 2 (5 Courses) 11
5.1.3 Year I Recess term 12
5.1.4 Semester 3 (5 Courses) 12
4.1.5 Semester 4 12
4.1.6 Year 2 Recess Term 12
4.1.7 Semester 5 12
4.1.8 Semester 6 13
6. Detailed Curriculum 13
6.1 Year 1 Semester 1 13
6.1.1 CSC 1100: Computer Literacy (4 CU) 13
6.1.2 BIS 1104: Communication Skills for IT (4 CU) 14
6.1.3 CSC 1104: Computer Organization & Architecture (4 CU) 16
6.1.4 CSC 1108: Individual Project I (4 CU) 17
6.1.5 CSC 1107: Structured Programming (3 CU) 17
6.2 Year 1 Semester 2 19
6.2.1 CSC 1214: Object Oriented Programming (4 CU) 19
6.2.2 MTH 2203: Numerical Analysis I (3 CU) 20
6.2.3 BIS 1204: Data and Information Management I (4 CU) 21
6.2.4 MTH 1203: Calculus I (4 CU) 21
6.2.5 BIS 1206: Systems Analysis and Design (4 CU) 23

6.3 Year I Recess Term 24
6.3.1 CSC 1304: Practical Skills Development (5 CU) 24
6.3.2 CSC 1303: Cisco Certified Network Associate (Audited) 24
6.4 Year 2 Semester 1 24
6.4.1 CSC 2100: Data Structures and Algorithms (4 CU) 24
6.4.2 BSE 2103: Computer Networks (4 CU) 25
6.4.3 BSE 2105: Formal Methods (4 CU) 26
6.4.4 CSC 2113: Software Engineering (4 CU) 27
6.4.5 MTH 3105: Discrete Mathematics (3 CU) 29
6.4.6 CSC 2114: Artificial Intelligence (3 CU) 30
6.5 Year 2 Semester 2 31
6.5.1 CSC 2200: Operating Systems (4 CU) 31
6.5.2 CSC 1209: Logic Programming (3 CU) 32
6.5.3 CSC 2209: Systems Programming (4 CU) 34
6.5.4 CSC 2210: Automata, Complexity and Computability (3 CU) 35
6.5.5 BIT 2207: Research Methodology (3 CU) 36
6.6 Year 2 Recess Term 37
6.6.1 CSC 2303: Field Attachment (5 CU) 37
6.7 Year 3 Semester 1 37
6.7.1 CSC 3110: User Interface Design (4 CU) 37
6.7.2 BAM 2102: Entrepreneurship Principles (3 CU) 38
6.7.3 CSC 3115: Advanced Programming (3 CU) 39
6.7.4 CSC 3112: Principles of Programming Languages (3 CU) 40
6.7.5 BIS 3100: Modeling and Simulation (3 CU) 41
6.7.6 CSC 3121: Computer Graphics (3 CU) 41

6.7.7 CSC 3118: Computer Science Project I (5 CU) 42
6.7.8 MTH 3107: Linear Programming (3 CU) 43
6.8 Year 3 Semester 2 44
6.8.1 BSE 2206: Data Communications (4 CU) 44
6.8.2 CSC 3205: Compiler Design (3 CU) 45
6.8.3 CSC 3211: Computer Science Project II (5 CU) 46
6.8.4 CSC 3207: Computer Security (3 CU) 47
6.8.5 BIS 3205: Data Warehousing and Business Intelligence (4 CU) 48
6.8.6 BSE 3202: Distributed Systems Development (4 CU) 49
6.8.7 CSC 3217: Emerging Trends in Computer Science (3 CU) 49
7. Resources and Infrastructure 50
7.1 Funds 50
7.2 Staff 50
7.3 Lecture Space 50
7.4 Computer Laboratories and Software 50
7.5 Library Services 51
8. Quality Assurance 51
8.1 Feedback from students enrolled 51
8.2 Class meetings 51
8.3 Use of ICT in availing lecture materials 51
8.4 Peer review 51
8.5 External examiners' reports 52
8.6 Industrial Training placement reports 52
8.7 Tracer studies 52
9. Staff in the Department of Computer Science 52

Bachelor of Computer Science

1. Background to the B.Sc Computer Science Program

The Bachelor of Science in Computer Science is the oldest undergraduate degree Programme in Makerere University School of Computing and Informatics Technology. It was launched at the inception of Institute of Computer Science in 2001. It was later revised in 2009. The latest revised curriculum will take effect in the 2013/2014 Academic Year. The objectives of the Programme are to:

1. Develop professionals with theoretical and practical skills in Computer Science;
2. Strengthen institutional capacity and building in Computer Science in tertiary institutions, the private and public sector;
3. Build capacity with a practical orientation needed to link up the Computer Science sector with Government and Industry under the broader perspective of Information and Communication Technology (ICT) and
4. Provide most of the ICT professionals needed in Uganda and neighboring countries.

1.1 Who is a Computer Science Graduate?

Our interdisciplinary approach to study sets us apart from traditional, highly structured computer science departments. As such, our students gain a strong foundation in computer science while preparing themselves to use their degree to solve real-world challenges.

The objective of the computer science program is to provide computer science graduates with a balanced breadth and depth of knowledge in computer science that allows them the choice between continuing their education in graduate school and beginning their professional career, and to excel in either environment.

Therefore the computer science graduate should be able to achieve the following outcomes:

- ability to apply knowledge of mathematics and science to carry out analysis of computer science problems and design appropriate solutions
- ability to use techniques, skills, and modern software development tools necessary for computing practice
- ability to identify, formulate, and solve computer science problems
- ability to design a computing system to meet desired needs
- ability to apply problem-solving strategies to new, unknown, or open-ended situations in computer science
- knowledge and understanding of the impact of the many sub-disciplines of computer science
- ability to function on teams
- ability to use written and oral communication skills effectively
- understanding of professional and ethical responsibility
- recognition of the need for and ability to engage in lifelong learning
- ability to acquire and use the ever-changing technical knowledge required of computing professionals

2. Rationale for reviewing the Programme

With the technology shifts and an economy that is based on different skills, we realized a need for revising the curriculum for our program so as to mitigate some of the obvious catastrophes and also provide the best
for our graduates. We understand that for us to achieve an effective curriculum revision, it requires a thorough understanding of the processes and principles of the changing paradigms affecting curriculum development.

In lieu of this, we considered various aspects in this process including:

- Cohort: requirement of national council to review

- Market analysis: We undertook a preliminary market analysis to evaluate the attractiveness and the dynamics of the Computer Science Market within East Africa. Special attention was paid on the kind of jobs/roles being advertised within this market segment and the different skills set that are being sought. We compared this with the Internship feedback obtained from our industrial partners where our students get posted. With this information, we were able to identify the opportunities, strengths, weaknesses and threats of our program.
- Industrial and Alumni feedback: We also held a workshop with our industrial partners to generate input into this process. In particular, key stakeholders and experts from regulatory authorities, industry and academia were invited to evaluate our current program. The feedback generated in this final phase of this process was very important in reconsidering certain content of the courses we have been offering.
- Student sample: A sizable sample of our students was also sampled. In here, we were testing their understanding of what we are offering against what they expected. These respondents pointed us to various areas which they considered a duplication of effort or redundancy in the program. Their feedback was thus important in helping us to eliminate duplicates and also for strengthening certain areas of our program.
- External examiners' and QA reports: These reports were also a great input into this review process. Although some of the comments given were majorly on assuring quality of students' learning experiences and also the fair assessment, other comments given to improve the program were very significant in highlighting areas which were found lacking in our program.

3. The Program

3.1 Target Group

The programme targets two categories of people. These are A' level leavers and diploma holders in relevant disciplines.

3.2 Nature of the Program

The program is full time that is conducted both in the day and in the evening.

3.3 Duration

The duration of the program shall be three academic years consisting of six semesters and two recess terms. Each semester lasts seventeen (17) weeks two of which are for examinations. Each recess term is 10 weeks.

3.4 Tuition Fees

Tuition fees for privately sponsored students shall be 3,024,000 Uganda Shillings per year for Ugandans and 4,536,000 Uganda Shillings per year for non-Ugandans.

4. Regulations

4.1 Admission requirements

To be admitted to the B.Sc (Computer Science) program, a candidate must satisfy the general admission requirements for Makerere University. In addition, the following regulations shall hold:

(a) Direct Entry

Candidates seeking admission through this avenue must have obtained:-

- At least a principle pass in Mathematics in the Uganda Advanced Certificate of Education (UACE) or its equivalent
- At least two principle passes at the same sitting in UACE in any of the following subjects: - Economics, Entrepreneurship, Geography, Physics, Chemistry and Biology.
- A minimum weighted point set by the Makerere University Admissions Board.

For purposes of computing weighted points, the A' level subjects shall be grouped and weighted as per the University weighting system.

Group	Weight	Subjects
Essential	3	Any two best done subjects among: Mathematics, Physics, Chemistry, Biology, Technicial Drawing
Relevant	2	Any other best done subject of all A' level subjects.
Desirable	1	General Paper, Subsidiary Mathematics, Subsidiary ICT
Others	0.5	All others.

(b) Diploma holders:

For a candidate to be admitted via the diploma scheme, he/she must:

i. Have at least 5 passes got at the same sitting of Uganda Certificate of Education or its equivalent, with credits in English and Mathematics
ii. Have at least 1 principal pass and 2 subsidiary passes from the same sitting of the Uganda Advanced Certificate of Education (UACE) or its equivalent
iii. Have a Diploma in Computer Science and Information Technology of Makerere University or have at least a Second class (lower division) diploma in Computer Science, Engineering, Information Technology, and Statistics or any other diploma with Mathematics, Computer Science, or Information Technology as one of the subjects. The diploma must be from an Institution recognized by the National Council for Higher Education in Uganda.

4.2 Progression

Progression shall be regarded as normal, probationary or discontinuation as per the standard of Makerere University Senate guidelines.

4.2.1 Normal Progress

This occurs when a student passes each course taken with a minimum Grade Point of 2.0

4.2.2 Probationary

This is a warning stage and occurs if either the cumulative grade point average (CGPA) is less than 2.0 and /or the student has failed a core course. Probation is waived when these conditions cease to hold

4.2.3 Discontinuation

When a student accumulates three consecutive probations based on the CGPA or the same core course(s), he /she shall be discontinued.

4.2.4 Retaking a Course

A student may re-take any course when it is offered again in order to pass if the student had failed the course. A student may take a substitute elective, where the student does not wish to re-take a failed elective.

4.3 Weighting System

The weighting unit is the Credit Unit (CU). The Credit Unit is a contact hour per week per semester. A contact hour is equal to (i) one lecture hour (LH), (ii) two practical hours (PH) or (iii) two tutorial hours (TH)

4.4 Semester Load and Minimum Graduation Load

The normal semester load is between 15 and 21 credit units. The minimum graduation load is 120 credit units of which 106 credit units are from core course units. The remaining 14 credit units are to be gotten from elective course units in semesters where elective courses are offered.

4.5 Course Assessments

(a) Each course will be assessed on the basis of 100 marks with proportions as follows:- Coursework – 40% and Examination 60%
(b) A minimum of two course assignments/tests shall be required per course
(c) Course work shall consist of individual tests, group assignment and presentations in each semester.

4.6 Grading and Pass mark

(a) Each course will be graded out of a maximum of 100 marks and assigned an appropriate letter grade and a grade point as follows:

Marks	Letter Grade	Grade Point	
90-100	A+	5	Exceptional
80- 89	A	5	Excellent
75- 79	B+	4.5	Very good
70- 74	B	4	Good
65- 69	C+	3.5	Fairly good
60- 64	C	3	Pass
55- 59	D+	2.5	Marginal fail
50- 54	D	2	Clear fail
45- 49	E	1.5	Bad fail
40- 44	E-	1	Qualified fail
0 - 39	F	0	Qualified fail

A student with a grade point greater or equal to 2 (letter grade D) in a certain course unit is considered to have passed the course unit.

(b) The following additional letters will be used, where appropriate:-
W - Withdrawal from course;
I – Incomplete;
AU – Audited Course Only
P – Pass;
F- Failure.

4.7 Minimum Pass Mark

A minimum pass grade for each course shall be 2.0 grade points

4.8 Calculation of Cumulative Grade Point Average (CGPA)

The CGPA shall be calculated as follows:

$$CGPA = \frac{\sum_{i=1}^{n} GP_i \times CU_i}{\sum_{i=1}^{n} CU_i}$$

Where GP_i is the Grade Point score of a particular course *i*; CU_i is the number of Credit Units of course *i*; and *n* is the number of courses so far done.

4.9 Knowledge Areas Covered in the Curriculum

The curriculum is based on 8 broad knowledge areas. These are:-

1. Mathematical Foundations of Computer Science (MFCS)
2. Theoretical Foundations of Computer Science (TFCS)
3. Data Structures, Algorithms and Programming (DSA&P)
4. Organizational Aspects of Software Development (OASD)
5. Study and Optimization of Operational Systems (SOOS)
6. Computer Networks (CN)
7. Research and Development (R&D)
8. Soft Skills (SS)

Graduates from the program are expected to exhibit competent knowledge/skills in all the subject areas above. Each of them, therefore, has to be adequately covered so as to produce good quality graduates. Analysis of the coverage for the different subject areas is done in Section 4.3.

4.9.1Content Distribution by Knowledge Area

The table below summarizes the distribution of the different course units in the different knowledge areas. The total number of credit units is also summarized.

	Knowledge Areas							
Sm	DSA&P	OASD	SOOS	CN	R&D	MFCS	TFCS	SS
I	CSC 3112		CSC 3112			MTH1203	CSC 1104	BIS 1104 CSC 1100
II	CSC 1205		CSC 1205		CSC 1208	MTH3105 MTH2203	CSC2210	
rtI				CSC 1303	CSC 1304			
III	CSC2116 BIS1204	CSC 2114 BIS1206		BSE2103				
IV	CSC 2209 CSC 2100 CSC1209	BSE 2201	CSC1209		BIT2207		CSC 2200 CSC 2214	
rtII					CSC 2303			
V	CSC 3115	CSC 3110 CSC3121	BIS3100		CSC 3118		CSC3117 BAM2102	
VI		BSE 2105 BSE2206 BSE 3202	BIS3205		CSC 3211	MTH3107	CSC 3207 CSC 3205	
∑	27	30	17	8	24	13	25	8

NB: Credits from audited courses are also included

From the table above, the largest proportion of the contact hours are in TFCS, OASD, R&D and DSA&P. This is aimed at having students with

(i) hands on skills

(ii) ability to research and learn new subject matter on their own and

(iii) ability to undertake advanced studies

5. The Curriculum

As a base for the curriculum, we have taken a "Computing Curricula Information Technology" by Association of Computer Machinery as well as looking at some of the content of UK universities like University of Surrey. The assumption here is that all the students admitted shall be computer literate but for those who are not there is a bridging course that is done in the first year.

5.1 Course Outline

Key:

1. **Modified**: Modified remark means that the course unit has been revised by content.
2. **Star on Course Code**: This means the course unit has been modified only by credit units and not by content.
3. **New**: Means the course unit has been freshly introduced in the department
4. **Old**: Means no change on the course unit's name, code and content

5.1.1 Semester 1 (5 Courses)

Code	Name	LH	PH	TH	CH	CU	Type	Remark	Mother Dept
Cores:- (5 Cores)									
CSC 1100	Computer Literacy	30	60	-	60	4	Core	Old	CS
BIS 1104	Communication Skills for IT	45	30	-	60	4	Core	Modified	IS
CSC 1104	Computer Organization & Architecture	60	-	30	45	4	Core	Old	CS
CSC 1108	Individual Project I	15	90	-	60	4	Core	Modified	CS
CSC 1107	Structured Programming	30	30	-	45	3	Core	Modified	CS
	Total Credit Units					**19**			

5.1.2 Semester 2 (5 Courses)

Code	Name	LH	PH	TH	CH	CU	Type	Remark	Mother Dept
Cores:- (5 Cores)									
BIS 1206	System Analysis and Design	45	-	30	60	4	Core	New	IS
MTH 1203	Calculus I	45	-	30	45	4	Core	Old	Math
CSC 1214	Object Oriented Programming	30	60	-	60	4	Core	Modified	CS
MTH 2203	Numerical Analysis I	45	-	30	45	3	Core	Modified	Math
BIS 1204	Data & Information Management I	30	60	-	60	4	Core	New	IS
	Total Credit Units					**19**			

5.1.3 Year I Recess term

Code	Name	LH	PH	TH	CH	CU	Type	Remark	Mother Dept
*CSC 1304	Practical Skills Development	15	90	-	75	5	Core	Modified	CS
CSC 1303	Cisco Certified Network Associate	150		-	75	5	Audited	Old	CS
	Total Credit Units					**5**			

5.1.4 Semester 3 (5 Courses)

Code	Name	LH	PH	TH	CH	CU	Type	Remark	Mother Dept
Cores:- (4 Cores)									
CSC 2100	Data Structures and Algorithms	45	-	30	60	4	Core	Old	CS
CSC 2114	Artificial Intelligence	30	30	-	45	3	Core	Old	CS
BSE 2103	Computer Networks	45	30	-	60	4	Core	New	NW
MTH 3105	Discrete Mathematics	30	-	30	45	3	Core	Old	Math
Electives:- (1 Elective)									
BSE 2105	Formal Methods	45	-	30	60	4	Elective	Old	NW
CSC 2113	Software Engineering	45	-	30	60	4	Elective	Old	CS
	Total Credit Units					**18**			

4.1.5 Semester 4

Code	Name	LH	PH	TH	CH	CU	Type	Remark	Mother Dept
Cores:- (5 Courses)									
CSC 2200	Operating Systems	45	-	30	60	4	Core	Old	CS
CSC 1209	Logic Programming	30	30	-	45	3	Core	Modified	CS
CSC 2209	Systems Programming	45	-	30	60	4	Core	Old	CS
CSC 2210	Automata, Complexity & Computability	45	-	-	45	3	Core	Old	CS
BIT 2207	Research Methodology	30	-	30	45	3	Core	Modified	IT
	Total Credit Units					**17**			

4.1.6 Year 2 Recess Term

Code	Name	LH	PH	TH	CH	CU	Type	Remark	Mother Dept
*CSC 2303	Field Attachment	-	300	-	75	5	Core	Modified	CS
	Total					**5**			

*Changed Name to University Name and CUs changed to 5

4.1.7 Semester 5

Code	Name	LH	PH	TH	CH	CU	Type	Remark	Mother Dept
Cores:- (4 Courses)									
CSC 3110	User Interface Design	45	30	-	60	4	Core	Modified	CS
BAM 2102	Entrepreneurship Principles	30	-	30	45	3	Core	Modified	Business
CSC 3112	Principles of Programming Languages	45	-	-	45	3	Core	Old	CS
CSC 3118	Computer Science Project I	-	-	150	75	5	Core	New	CS
Electives :- (1 Elective)									
*BIS 3100	Modeling and Simulation	30	30	-	45	3	Elective	Modified	IS

CSC 3121	Computer Graphics	30	30	-	45	3	Elective	Old	CS
CSC 3115	Advanced Programming	45	30	-	45	3	Elective	Old	CS
MTH 3107	Linear Programming	30	-	-	30	3	Elective	Old	Math
	Total Credit Units					**18**			

4.1.8 Semester 6

Code	Name	LH	PH	TH	CH	CU	Type	Remark	Mother Dept
Cores:- (3 Cores)									
BSE 2206	Data Communications	45	30	-	60	4	Core	New	NW
CSC 3205	Compiler Design	45	30	-	45	3	Core	Old	CS
CSC 3211	Computer Science Project II	-	-	150	75	5	Core	Modified	CS
Electives :- (2 Electives)									
CSC 3207	Computer Security	45	30	-	45	3	Elective	Old	CS
BIS 3205	Data Warehousing & Business Intelligence	45	30	-	60	4	Elective	Modified	IS
BSE 3202	Distributed Systems Development	45	30	-	60	4	Elective	Old	NW
*CSC 3217	Emerging Trends in Computer Science	45	45	-	45	3	Elective	Modified	CS
	Total					**19/20**			

6. Detailed Curriculum

6.1 Year 1 Semester 1

6.1.1 CSC 1100: Computer Literacy (4 CU)

(a) Description

In this course, students are to learn about the basic organization, concepts and terminologies in a computerized environment. They are also to get an in depth understanding of common computer applications. The use of related applications in different operating systems will be explored.

(b) Aims

The aims of the course unit are to:

- Equip students with basic knowledge about computer organization;
- Equip students with skills of using common office applications;
- Expose students to different operating systems;
- Equip students with skills of how to use the Internet and
- Equip students with knowledge about common text editors in different operating systems.

(c) Learning outcomes

By the end of the course unit, the student should be able to:

- Describe the different parts of a computer;
- Describe the historical evolution of computers;
- Competently use the common office applications in at least two operating systems;
- Competently use common text editors in at least two operating systems.

(d) Teaching and learning pattern

Teaching will be by lectures and laboratory demonstrations/practicals

(e) Indicative content

- General computer organization
- Historical perspectives of computing
- Common Microsoft office packages
- Office packages in other operating systems
- Text editors
- Common Linux commands
- Using the web

(f) Assessment method

The assessment will be in form of tests and assignments (40%) and final written exam (60%)

(g) Reading List

(i) Computer Literacy by John Preston, Robert Ferrett and Shelly Gaskin, 2007.
(ii) Practical Computer Literacy by Jelne Janrich and Dan Oja, 2001
(iii) Computer Literacy for IC^3 Unit 1: Computer Fundamentals QA 76.3 P743 2010
(iv)Computer Literacy for IC^3 Unit 2: Using productivity Software QA 76.3 P743 2010
(v) Computer Literacy Basics A Comprehensive Guide to IC^3 QA76.3 M67 2010

6.1.2 BIS 1104: Communication Skills for IT (4 CU)

(a) Description

This course provides students with skills of effective communication. Emphasis is provided on communications in today's business environment that has increasingly been shaped by Information technology. Students will be taught how to effectively communicate technical information to lay audiences using oral, written and non-verbal communication.

(b) Aims

The aims of the course are:

- Ensure that students develop and apply effective communication and writing skills in management of information systems.
- To familiarize students with the concept and Effective communication, reports and business documents required effectively in the management of information systems.
- To teach students the principles of designing formal and informal reports to meet the needs of a variety of information system managers.
- To enable students understand the importance of effective organization and personal communication.
- To enable students prepare some business documents in the organization / institutions.
- To provide tools and strategies with which students can systematically improve their communication skills.

(c) Learning outcomes

By the end of the course unit, the student should be able to:

- Identify, explain, and demonstrate different types of communication techniques.
- Organise and present ideas individually and as part of a team.
- Demonstrate interpersonal skills including business etiquette, active listening, team participation, and leadership skills.
- Identify ways of communicating effectively through an appropriate balance between interpersonal and technical capabilities.
- Produce a text-based report or proposal, using Microsoft Word.
- Apply a universal process that enables you to solve communication problems now and throughout your entire carrier

(d) Teaching and Learning Pattern

This is a very practical course. Students will be required to real life examples and illustrations to enable them apply knowledge with the business context.

(e) Indicative Content

- Understanding the Foundations of Business Communication
    - Effective Business Communication 3
    - Communicating in Teams and Mastering Listening and Nonverbal Communication
- The Writing Process
    - Planning Business Messages
    - Writing Business Messages
    - Completing Business Messages
- Crafting Brief Messages
    - Crafting Messages for Electronic Media
    - Writing Routine and Positive Messages
    - Writing Negative Messages
    - Writing Persuasive Messages
- Planning, Writing, and Completing Reports and Proposals
- Designing and Delivering Oral and Online Presentations
    - Creating and Delivering Oral and Online Presentations
    - Enhancing Presentations with Slides and Other Visuals
    - Writing Employment Messages

(f) Assessment Method

Students should be encouraged to analyse business domain problem and come up with communications to address the needs as part of their coursework assessment (20%). Other assessments include test (20%) and exams (60%).

(g) Reading List

(i) COURTLAND L. BOVÉE and JOHN V. THILL, Business Communication Today. 8th edition. Upper Saddle River, NJ: Prentice Hall International Inc. (Call No. HF5718 Bov2005)

(ii) Locker, Kitty O. 2006. Business and Administrative Communication. 7th edition. Boston, Mass.: Irwin/McGraw-Hill(Call No. : HF5718 Loc2006)

(iii) Mastering Business communication By L. A Woolcott & W.R Unwin, Macmillan Publishing House.

(iv) Effective Business communication by Asha Kaul, Printice of India

(v) Business & Administrative communication by Kitty O. Locker (1995) 3rd Edition.

(vi) Effective Business communication. Getting the message around by Waswa Balunywa and Ngoma Muhammad

6.1.3 CSC 1104: Computer Organization & Architecture (4 CU)

(a) Description

This course introduces the logical architecture and organization of computer systems. It highlights the lower end operations in a typical computer as well as the way computers manage their resources during operation. The course opens up a student to be an informed user of the computer rather than a passive recipient of the computer services.

(b) Aims

The aims of the course are:

- To introduce to students the concepts of computer organization;
- To highlight to students the way computers process and store the data;
- To highlight internal management issues in computer systems.

(c) Learning Outcomes

By the end of the course unit, the student should be able to:

- Explain the difference between Computer Organisation and Computer Architecture;
- Discuss how information is processed in the computer from the time it is input to the time of output;
- Explain how the Micro processers operate;
- Explain the .difference between the various representation codes i.e. ASCII and EBCDIC

(d) Teaching and Learning Pattern

Teaching will be in terms of lectures as well as tutorials

(e) Indicative Content

- Data Representation:
  Integer Formats, Binary, Octal and Hexadecimal Systems, Negative integers and 2's Complement, Floating Point Formats, BCD Formats, Alphanumeric Memory, Memory Management Hardware Codes.
- Basic Digital Circuits:
  Logic gates, Karnaugh maps, Combinatorial Circuits, Binary Adders, Multiplexers and Demultiplexers, Comparators, Decoders and Encoders, Code Converters, ROMS and PLA's, Sequential Circuits, Flip Flops and Latches, R-S flip flops, J-K flip flops, T flip Flops, D flip flops, Registers, Shift Registers and Data Transmission, Sequential Network Design.
- Micro Computer Architecture:
  CPU, Memory, I/O Devices and Interfaces, System Bus, Examples of CPU Structures, The Intel / Pentium CPU, The Z-80 or Motorolla, Machine Language Instructions, Instruction Formats and Addressing Modes.
- The Processing Elements: Macroinstruction execution, Internal Bus Transfers, Detailed Internal Architecture, Microcontrol, Hardwired Control, Microprogrammed Control, Reduced Instruction Set Computers.
- I/O Programming:
  Programmed I/O, Interrupt I/O, Polling, Priority Interrupt System, Direct Memory Access Memory Management: Memory Hierarchy, Main Memory, I/O processors.
- Memory Systems and, Auxiliary Memory, Associative Memory, Cache Memory, Virtual

(f) Assessment method

Assessment will be in form of tests and assignments (40%) and final examination (60%)

(g) Reading List

(i) Computer Systems Architecture by M. Morris Mano, Prentice Hall, 1993

(ii) Structured Computer Organization by Andrew S. Tanenbaum, Prentice Hall 1984.

(iii) Computer Systems Concepts and Design by Glenn B. Gibson, Prentice Hall, 1991
(iv) Computer Organization and Architecture by William Stallings, Prentice Hall 2003.
(v) Computer Architecture and Organization by J.Hayes McGraw Hill

6.1.4 CSC 1108: Individual Project I (4 CU)

(a) Description

This course is to give students an experience of developing simple but complete applications. Focus will be put on programming and program documentations as well as realization of the objectives.

(b) Aims

The aims of the course are:

- To practically give students skills of integrating different concepts of programming into a single application.
- To introduce the student to hands on aspects of software development.
- To nurture the student's ability to independently read different sources of literature so as to identify what (s)he can use to develop his/her application.

(c) Learning outcomes

By the end of the course unit, the student should be able to:

- Apply the skills he /she has acquired to integrate different concepts of programming into a single application;
- Apply the skills he / she has acquired to software development;
- Solve real life problems and challenges

(d) Teaching and Learning Pattern

Learning will be largely by self - study/ research. Students will be given programming Assignments that will be submitted after a specific period of time. The skills which that will be expected in the assignment will be indicated to the student. A member of staff will meet students at the issue of each of the assignment and address any outstanding issues as well as expectations. A minimum of six assignments will be given.

(e) Indicative content

The main content in the course is translation of a real life problem into a working computer program.

(f) Assessment method

Two assignments (each taking two weeks) and three assignments (each taking 3 weeks) will be given. The first two assignments will constitute the coursework (40%) while the last three assignments will constitute the examination mark (60%).

(g) Reading List

Students can read any literature like books and online tutorials that can help in addressing the problem in the project at hand.

6.1.5 CSC 1107: Structured Programming (3 CU)

(a) Description

The course is to create a strong base in the principles and practice of functional programming. A high level programming language like C is to be used. Students are to cover both theoretical principles and hands on practical skills. The main concepts to cover include program structure, data structures, syntactical and semantic correctness, planning and segmentation in programming as well as working with files.

(b) Aims

The aims of the course are to provide the student with:

- Comprehensive knowledge about structured oriented programming;
- Knowledge in planning and organization of programming projects;
- Knowledge and techniques of evaluating syntactic and semantic correctness of a computer program;
- Strong practical basis in programming.

(c) Learning Outcomes

Students who successfully complete this course of study will be able to:

- Understand the basic terminology used in computer programming
- Write, compile and debug programs in C language
- Use different data types in a computer program
- Design programs involving decision structures, loops and functions
- Explain the difference between call by value and call by reference
- Use different data structures and create/update basic data files

(d) Teaching and Learning pattern

The course will be taught with a big practical component. Students will be expected to have one supervised practical sessions per week. They will so be given several programming assignments some of which will be marked and contribute to the coursework scores.

(e) Indicative content

- Program structure
- Variables and Operators
- Conditional statements
- Looping statements
- Arrays and strings
- Functions
- Advanced data types
- Pointers
- Dynamic memory allocation and dynamic structures
- Working with files
- GUI

(f) Assessment method

Assessment will be in form of at least one (practical) assignment and one test (40%), a practical exam - (30%) and a final written examination (30%)

(g) Reading List

(i) Brian W. Kernighan, Dennis M. Ritchie: C Programming Language; Prentice Hall, 2000.
(ii) Samuel P. Harrison: C- A Reference Manual (5th Edition); Prentice Hall, 2002.
(iii) Keith P. LaBudde: Structured programming concepts; McGraw Hill 1987
(iv) Donald Ervin Knuth: The Art of Computer Programming; McGraw Hill 2006
(v) K.N King: C Programming- A modern Approach

6.2 Year 1 Semester 2

6.2.1 CSC 1214: Object Oriented Programming (4 CU)

(a) Description

The course is to give an in depth understanding of Object Oriented programming. It is to cater for Object Oriented programming practices like inheritance, interfaces, exception handling, action handling, security, software reuse and robustness.

(b) Aims

The aim of the course is to

- Move the students' programming skills from basic to advanced
- Avail students with skills to handle non- functional program aspects like robustness and security
- Train students to develop complete computer applications

(c) Learning outcomes

Students who successfully complete this course of study will be able to:

- Explain the principles of the object oriented programming paradigm specifically including abstraction, encapsulation, inheritance and polymorphism
- Use an object oriented programming language, and associated class libraries, to develop object oriented programs
- Design, develop, test, and debug programs using object oriented principles in conjuncture with an integrated development environment
- Construct appropriate diagrams and textual descriptions to communicate the static structure and dynamic behaviour of an object oriented solution
- Describe and explain the factors that contribute to a good object oriented solution, reflecting on your own experiences and drawing upon accepted good practices.

(d) Teaching and learning pattern

This will include lectures, practicals and lab assignments

(e) Indicative content

- The object oriented paradigm
- Classes and objects
- Inheritance and visibility modifiers
- Interfaces and abstract classes
- Graphical user interface and action handlers
- Exception handling
- Working with files
- Working with databases
- Sessions and user management

(f) Assessment method

The assessment will be done by tests and take home assignments (40%), practical examination (30%) and written examination (30%)

(g) Reading List

(i) Java Software Solution: Foundations of Program Design (6th Edition) by John Lewis and William Loftus, Addison-Wesley, 2009.

(ii) A Programmer's Guide to Java TM Certification: A comprehensive Primer by Khalid A. Mughal and Rolf W. Rasmussen, Addison-Wesley, 1999
(iii) Java Programming by Kim N King McGraw Hill 2000
(iv) Java Enterprise in a Nutshell by David Flanagan
(v) Problem solving with Java by Elliot Koffman

6.2.2 MTH 2203: Numerical Analysis I (3 CU)

(a) Description

The course is to sharpen the students' skills in using numerical approaches to solving mathematical/real life problems. The focus will be on the ability to correctly formulate numerical problems and schemes that solve them. Emphasis will be put on the precision and robustness of the schemes.

(b) Aims

The aims of the course are

- To provide a solid basis on the numerical approaches to computational problem solving;
- To provide students with problem analysis and solving skills to be able to handle typical computational problems in practice.

(c) Learning Outcomes

By the end of the course, the student should be able to

- Derive schemes for different set ups of numerical problems
- Test for convergence of different schemes
- Correctly use the schemes to generate solutions to the numerical problems to the required precision

(d) Teaching and Learning Pattern

The course will be taught theoretically. The developed schemes can be implemented and run in any programming language.

(e) Indicative Content

- Errors and their propagations
- Langrange's Interpolating Polynomial
- Iterated interpolation
- Finite difference interpolating polynomials
- Numerical Differentiation
- Numerical Integration
- Adoptive iteration
- Solutions to non- linear equations
- Systems of non -linear equations
- Systems of linear equations

(f) Assessment method

At least 2 (1 hour) tests and 1 assignment (40%) 3-hour examination (60%)

(g) Reading List

(i) Numerical Analysis by Richard L. Burden and J. Douglas Faires, Wardsworth, 1993.
(ii) A.K. Kaw, E.E.Kalu and D.Nguyen(2008) Numerical Methods with Applications.
(iii) Numerical Analysis by Timothy Sauer, 2nd Edition, 2011
(iv) Applied Numerical Analysis using MATLAB, 2nd Edition by Laurene Fausett, May 2007
(v) Numerical Methods using MATLAB 4th Edition by Johm Mathews, Kurtis Fink

6.2.3 BIS 1204: Data and Information Management I (4 CU)

(a) Description

The course is 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.

(b) Aims

The aims of the course are to:

- Provide a background for the evolution of database (management) systems
- Provide the students with the steps one has to go through when developing good database applications
- Give hand on experience and knowledge in developing database (driven) applications

(c) Learning Outcomes

By the end of the course, the student should be able to:

- Describe the systematic approaches to design and implementation of database applications
- Describe the concepts of requirements gathering and database planning
- Discuss the development of application programs that talk to the database.

(d) Teaching and Learning patterns

Teaching will be by lectures, take home reading assignments/class presentations and laboratory practicals

(e) Indicative content

- Background to databases
- Evolution of database systems
- Database organization and architecture
- Database models
- The database development life cycle
- Database design
- Tuning of operational systems
- Querying databases
- SQL/PL SQL
- Scripting

(f) Assessment Methods

Assessment will be in terms of tests and take home assignments (40%), a practical examination (30%) and a final written exam (30%)

(g) Reading List

(i) Thomas Connolly and Carolyn Begg: Database Systems: A Practical Approach to Design, Implementation and Management. 2nd Edition, Addison-Wesley, 2004.

6.2.4 MTH 1203: Calculus I (4 CU)

(a) Description

The course gives the students a strong mathematical base to be able to tackle other computer problems. The course provides techniques that commonly used in the general area of computer science. It builds a foundation for other courses that need special mathematical backgrounds.

(b) Aims

The aims of the course are:

- To provide students with a mathematical base that is to be used to solve computer science problems
- To improve the problem solving skills of students

(c) Learning outcomes

By the end of the course unit, the student should be able to:

- Describe functions like composite, logarithmic, exponential and hyperbolic;
- Apply techniques and theorems of evaluating Limits;
- Explain the rules and theorems of determining derivatives;
- Describe the fundamental theorem of Calculus.

(d) Teaching and learning pattern

The teaching will largely involve lectures, together with tutorials and take home assignments.

(e) Indicative content

- Functions
  - ✓ Polynomials (linear, quadratic) rational, composite functions
  - ✓ Transcendental functions: logarithmic and exponential functions;
  - ✓ The trigonometrically functions and their inverses
  - ✓ The hyperbolic functions and their inverses
- Limits
  - ✓ Informal definition of limits of functions and continuity;
  - ✓ One sided limits
  - ✓ Removable discontinuity
  - ✓ Techniques and theorems of evaluating limits
  - ✓ Formal definition of limits
  - ✓ Application to definition and properties of continuous functions
  - ✓ Use of the definition in proofs and problem of limits and continuity
- Differentiation
  - ✓ Definition of a derivative, continuity and differentiability
  - ✓ Rules and theorems of determining derivatives
  - ✓ Inverse functions: their derivatives and graphs
  - ✓ Differentials: applications to approximation. Rolle's theorem, Mean Value Theorem, L'Ohopital's Rule
  - ✓ Anti-derivatives: Techniques and theorems for determining anti derivatives
  - ✓ Integration: Define Intergral, Rieman sums, the definite intergral and area,
  - ✓ The fundamental theorem of calculus: application to evaluation of definite intergrals (by substitution)
  - ✓ Functions defined by integration: f(t) dt as an anti -derivative of f(x), mean value theorem for intergrals

(f) Assessment method

The assessment will be by tests (20%), group assignments and presentations (20%) and final examination (60%)

(g) Reading List

(i) Calculus and Analytic Geometry (9th Edition) by George B. Thomas, Ross L. Finney Addison Wesley, 1995.
(ii) Introduction to Statistics by R. E Walpole, 3rd Ed, Prentice Hall, 1982.
(iii) Calculus by Hughes-Hallett, Gleason, McCallum Wiley
(iv) Calculus in Context by James Callahan, et al, 2008
(v) Reform Calculus by Marcel B. Finan, Arkansas Tech University, 2004

6.2.5 BIS 1206: Systems Analysis and Design (4 CU)

(a) Description:

This course introduces established and evolving methodologies for the analysis and design of an information system. Great emphasis is placed on system characteristics, managing projects, prototyping, CASE/OOM tools, systems development life cycle phases, the role of the systems analyst, systems selection, definition of systems requirements, feasibility analysis, system design, and system architecture are topics included.

(b) Aims

To introduce software engineering and to explain its importance
- To set out the answers to key questions about software engineering
- To introduce ethical and professional issues and to explain why they are of concern to software engineers

(c) Learning Outcomes

By the end of the course, the student should be able to
- Describe the established and evolving methodologies for the analysis and design of an information system
- Describe the systems development life cycle phases
- Describe the role of the systems analyst, and systems selection
- Analyze a problem and design an appropriate solution using a combination of tools and techniques.

(d) Teaching and learning pattern

Teaching and learning is to be implemented through lecture, lab and tutorial sessions. Students are also expected to make presentations of their work.

(e) Learning outcomes

On successfully completing of this unit students will be able to
- Demonstrate competence in handling software engineering projects.
- Know the fundamental software engineering processes and models
- Know what is involved in a typical software engineering project's life cycle.
- Employ good project management principles in handling projects and know why these principles are important in constructing quality software.
- Be competent in using CASE tools in real world projects.

(f) Indicative Content:

- SAD Fundamentals: Introduction to IS & types of IS; Need for SAD & role of Analyst; SDLC & use of CASE tools; Determining feasibility & mgt of SAD activities.
- Information Requirements Analysis Information gathering (Interactive methods, unobtrusive methods, RAD, prototyping); Determining Systems requirements (types of requirements)
- The Analysis Process: Structuring systems requirements (describing process specifications & structured decisions) Data modelling; Process modelling; Preparing System proposals.
- The Design Process: Designing effective output; Designing effective input (Accurate Data entry Procedures); User Interface design; Database Design.
- System Implementation: Quality Assurance through Software Engineering (design with structured charts, testing, maintenance, auditing, Quality mgt); Implementing Information Systems (user training, conversion strategies, systems evaluation.

- Project Management: Stages of system Development; Project planning; Estimation & Project Monitoring & Control.
- Introduction to Object-oriented Systems Analysis & Design using UML

(g) Reading List

(i) Kendall & Kendall, *Systems Analysis and Design,* 6th edition, Pearson Prentice Hall, 2005.

(ii) A. J. Hoffer, F. J. George and J. S. Valacich, *Modern Systems Analysis and Design,* 2nd edition, Addison-Wesley, 1999.

6.3 Year I Recess Term

6.3.1 CSC 1304: Practical Skills Development (5 CU)

The course aims at imparting practical skills in areas chosen by the faculty. The students are to be supervised by staff with in the faculty. Areas of practical skills development include

- Implementation of projects
- Network and system administration
- Hardware maintenance
- Computer assembly

This can be done within the School of Computing and Informatics Technology or any other unit in Makerere University. Students will write a report at the end of the course.

6.3.2 CSC 1303: Cisco Certified Network Associate (Audited)

In this course, students will cover the course content of the CCNA international curriculum.

6.4 Year 2 Semester 1

6.4.1 CSC 2100: Data Structures and Algorithms (4 CU)

(a) Description

The course gives students a firm foundation of data structures and algorithms. The course trains students on systematic development and analysis of algorithms. The importance of algorithm complexity on computer performance is emphasized. Typical computational problems and their solutions/analysis are to be covered.

(b) Aims

The aims of the course are to

- Make students appreciate the role of data structures and algorithms in computer programs;
- Improve students' problem solving skills by subjecting them to step by step analysis and design of computer algorithms;
- Introduce students to concepts Data structures;
- Introduce students to concepts of algorithm analysis;
- To expose students to generic algorithmic problems and apply them to other computational scenarios.

(c) Learning Outcomes

By the end of the course, the student should be able to

- Appreciate the role of data structures and algorithms in computer programs
- Describe a step by step analysis and design of computer algorithms
- Analyze generic algorithmic problems and apply them to computational scenarios

(d) Teaching and Learning pattern

The teaching pattern is by lecture, practical lab work, group discussion and class presentations.

(e) Indicative content

- Complexity analysis (Big-O notation, orders of growth, worst case, average case and amortized analysis);
- NP-complete problems;
- Greedy algorithms;
- Dynamic programming;
- Design patterns for data structures;
- Graphical representation of optimization problems
- Parallel algorithms.
- Sorting and searching;
- Divide-and-conquer algorithms;
- Elementary data structures;
- Recursive data structures (stacks, queues, linked lists, trees);
- Storing and searching (hash tables, search trees);
- Graph algorithms on graphs (shortest path, spanning trees).

(f) Assessment method

The assessment will be done by tests/assignment (40%) and final examination (60%)

(g) Reading List

(i) Data Structures and Algorithms by Alfred V. Aho, Jeffrey D. Ullman, John E. Hopcroft. Addison-Wesley, 1983
(ii) The Design and Analysis of Computer Algorithms by Alfred V. Aho, Addison-Wesley Longman, 1974
(iii) Introduction to Algorithms 2nd Ed by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, McGraw-Hill, 2008.
(iv) Problem Solving with Algorithms and Data Structures Using Python, by Brad Miller, David Ranum, 2011
(v) Data Structures and Algorithm Analysis in Java, by Clifford A. Shaffer - Dover Publications , 2012

6.4.2 BSE 2103: Computer Networks (4 CU)

(a) Description

This course examines principles, design, implementation, and performance of computer networks and data communication. The aim is to sharpen student's understanding and skills in computer networking and data communication. Subjects for discussion include: Internet protocols and routing, local area networks, wide area networking, wireless communications and networking, performance analysis, congestion control, TCP, network address translation, multimedia over IP, switching and routing, mobile IP, peer-to-peer networking, network security, and other current research topics and technologies.

(b) Aims

The aims of the course are

- To provide a solid basis on the theoretical and practical understand of data communication networks
- To introduce students to standards and guidelines in computer and data communication networks
- To impact knowledge and skill relevant for the design, implementation and maintenance of modern computer communication networks
- To introduce students to emerging technologies in data communication

(c) Learning outcomes

Upon successful completion of this course, the student should be able to

- Describe theoretical concepts of computer communication systems, including standards, protocols, and infrastructure devices e.g. routers and switches.
- Design, install and manage a simple Local Area Network or A campus Wide Area Network
- Describe emerging communication technologies particularly wireless and fiber optic technologies
- Deploy sound network security practices and tools
- Describe emerging research directions in computer communication networks
- To write a few network scripts

(d) Teaching and Learning Pattern

The course will be delivered inform of lectures, tutorials, lab experimentation, and group assignments.

(e) Indicative Content

- Network services and applications: DNS, HTTP, SMTP, peer-to-peer systems
- Network transport architectures, TCP, UDP, TCP congestion control
- Routing and forwarding, intra-domain, inter-domain routing algorithms and Mobile IP
- Link layers and local area networks, Ethernet, WiFi, and mobility
- Multimedia communications and quality of service
- Network measurement, inference, and management
- Network security (ACL, IPSec, etc)
- Network programming
- Network experimentation and performance analysis
- Protocol verification

(f) Assessment method

Assessment will be in terms of tests and Assignment (40%) and final examination (60%)

(g) Reading List

(i) James F. Kurose and Keith W. Ross. Computer Networking - A Top Down Approach Featuring the Internet, 3rd edition, Addison-Wesley, 2004, ISBN 0-321-22735-2.

(ii) L. Peterson and B. Davie, Computer Networks: A Systems Approach. Morgan Kaufmann Publishers, 1999.

6.4.3 BSE 2105: Formal Methods (4 CU)

(a) Description

The course provides students with skills of solving generic formal problems in science. It covers the intellectual and practical skills necessary for problem formalization.

(b) Aims

The aims of the course are:

- To provide students with factual knowledge including the mathematical notations and terminologies used in formalizing scientific problems
- To provide students with fundamental principles including the laws and theorems arising from the concepts covered in this course;
- To be able to apply course material along with techniques and procedures to solve practical problems;
- To provide programming skills by writing numerical programs like Matlab programs, to solve numerical problems.

(c) Learning Outcomes

By the end of the course, the student should be able to

- Describe the mathematical notations and terminologies used in formalizing scientific problems
- Discuss the fundamental principles including the laws and theorems arising from the concepts covered
- Apply numerical programs like Matlab programs, to solve numerical problems.

(d) Teaching and Learning Patterns

Teaching will be by Lectures and practical demonstrations

(e) Indicative Content

- Predicate Logic Specification:
  Foundations; Basic concepts; Verification; Z; Tools and systems; Z animation Miranda and ZANS; Nitpick and the Z Notation.
- Algebraic Specification:
  Foundations; Basic concepts; Verification; Tools and systems; Miranda; The OBJ family of languages; LARCH.

(f) Assessment Method:

Assessment will be in terms of tests and practical assignments (40%) and final written examination (60%)

(g) Reading list

(i) Z: An Introduction to Formal Methods, by Antoni Diller, 2nd edition, Wiley, (June 1994), ISBN-10: 0471939730

(ii) Logic in Computer Science: Modeling and Reasoning about Systems, by Michael Huth and Mark Ryan, Cambridge University Press; 2nd Edition (August, 2004), ISBN-10: 052154310X

(iii) Formal Methods and Models for System Design: A System Level Perspective, by Gupta, R., Le Guernic, P. Shukla, S.K. and Talpin, J.P. (Eds.), 2004, ISBN: 978-1-4020-8051-7

6.4.4 CSC 2113: Software Engineering (4 CU)

(a) Description

This course introduces students to the foundations of software engineering as a discipline. Students are introduced to the evolving role of software engineering, especially with emphasis on software engineering process and process models. Key topics covered include Software configuration management, Requirement analysis, Software Specification, Design methods, Software testing, Software project management techniques; Software project planning, Risk management; Software Quality Assurance; Software reuse; and Computer aided software engineering: CASE tools and application.

(b) Aims and Objectives

• To introduce software engineering and to explain its importance

• To set out the answers to key questions about software engineering

• To introduce ethical and professional issues and to explain why they are of concern to software engineers

(c) Learning outcomes

On successfully completing of this unit students will be able to

• Demonstrate competence in handling software engineering projects.

• Know the fundamental software engineering processes and models.

• Know what is involved in a typical software engineering project's life cycle.

• Employ good project management principles in handling projects and know why these principles are important in constructing quality software.

• Be competent in using CASE tools in real world projects.

(d) Teaching and Learning Pattern

Teaching and learning is to be implemented through lecture, lab and tutorial sessions. Students are also expected to make presentations of their work.

(e) Indicative Content

• Evolving role of software, software characteristics; Systems and environment; system engineering hierarchy, information and knowledge engineering; Information strategy; Business Area analysis, modeling enterprise and business-level data modeling, system architecture and associated information flow; writing system specification.

• Software Engineering as a layered technology: Software process, software process models. Software configuration management: the SCM process, Identification of objects in software configuration, version control, change control, configuration audit, SCM standards.

• Requirement analysis: Communication techniques, Information gathering tools; organizing and structuring information; analysis principles; Analysis modeling.

• Software Specification: Design process, principles and concepts: Abstraction, refinement, modularity, control hierarchy, structural partitioning, information hiding, functional independence, cohesion, coupling, design heuristics;

• Design methods: data design, architectural design, transform mapping, design optimization, human computer interface design, procedural design and tools; Design documentation.

• Software testing: Testing objectives, Testing principles, Testability, test case designing, white box testing; Basis path testing: Condition testing, data flow testing, loop testing; Black box testing: graph based testing methods, equivalence partitioning, Boundary value analysis, comparison testing; Testing documentation and help facilities; Software testing strategy: unit testing, integration testing, validation testing, system testing.

• Software project management techniques: project metrics, software measurement and metrics, software quality metrics;

• Software project planning: objectives of planning, resources, project estimation and estimation models, project decomposition techniques, make-buy decisions; automated estimation tools.

• Risk management: software risks, risk identification, risk projection, risk mitigation, monitoring and management; Project Scheduling: people and effort relationships, defining tasks, defining task network, scheduling techniques; Software teams and intra-team relationships; role of project manager.

• Software Quality Assurance: Concept of quality, quality control vs. quality assurance, cost of quality, factors that affect quality, quantitative view of quality, quality metrics, defect removal efficiency SQA activities, ISO standards and CMM practices, SEI levels, Software reviews, Formal approaches to SQA, Statistical Quality Assurance. Software reliability, reliability metrics, reliability models, meeting reliability requirements.

• Effective metrics for software process: Measurement principles, attributes of software metrics, metrics for analysis model, metrics for design model, metrics for source code, metrics for maintenance.

• Software reuse: difficulties in reuse, hardware reuse vs. software reuse, reusable artifacts, domain engineering approach, analysis design and construction of reusable components, classification and retrieval of components, economic impact of reuse and reuse metrics.

• Computer aided software engineering: CASE tools and application.

(f) Assessment method

• Continuous assessment through practical exercises and Coursework, together with two scheduled tests (40%)

• Final exam at the end of the Semester and accounts for 60% of the final grade.

(g) Reading lists

(i) J.F. Peters, W.Pedrycz,Software engineering: An Engineering approach, John Wiley, 2000.

(ii) R.S. Pressman, Software engineering: A Practitioners Approach, 5th Edition, McGraw Hill, 2005.

(iii) Sommerville, Software engineering, 8th Edition, Addison Wesley, 2008.

(iv) D. Ghezzi, M. Jazayeri, D. Mandrioli, Fundamentals of software Engineering, Prentice Hall of India, 2004.

6.4.5 MTH 3105: Discrete Mathematics (3 CU)

(a) Description

The course applies mathematics to finite or discontinuous quantities in order to master the process of problem- solving, communication, reasoning, and modeling. It gives a basic understanding of mathematical structures that are fundamentally discrete. Objects studied in discrete mathematics are largely countable sets such as integers, finite graphs, and formal languages. Applications of such concepts to computer science are to be studied. Concepts and notations from discrete mathematics are useful in studying and describing objects and problems in computer algorithms and programming languages.

(b) Aims

The aim of this course is to provide the student with: -

- A basic understanding of mathematical objects that assume only distinct, separate values, rather than values on a continuum.
- The main ideas studied in the broad area of Discrete Mathematics especially clear algorithmic aspects.
- An understanding of what the relation between problems is.

(c) Learning outcomes

By the end of the course unit, the student should be able to:

- Describe the process of problem solving, communication, reasoning and modeling;
- Discuss the applications of the concepts of integers, finite graphs and formal languages;
- Describe objects and problems in computer algorithms and programming languages.

(d) Teaching and Learning pattern

Teaching and learning will be by lectures and tutorials

(e) Indicative content

- Logics and set theory
- Number theory
- Relations and functions
- Languages
- Finite state machines (and optionally Finite state automata)
- Groups and Modulo Arithmetic
- Number of solutions of a linear equation
- Recurrence relations
- Searching algorithms

(f) Assessment method

Assessment will be by assignments and/or tests (40%) and written examination (60%)

(g) Reading List

(i) Bobrow, L.S. and Arbib, M.A. Discrete Mathematics: Applied Algebra for Computer and Information Science. Philadelphia, PA: Saunders, 1974.

(ii) Dossey, J.A.; Otto, AD.; Spence, L.; and Eynden, C.V. Discrete Mathematics, 3rd ed. Reading, MA: Addison-Wesley, 1997.

(iii) Balakrishnan, V.K. Introductory Discrete Mathematics. New York: Dover, 1997.

(iv) Kenneth H. Rosen; Discrete Mathematics and its Applications

(v) Johnsonbaugh; Discrete Mathematics 6th Edition

6.4.6 CSC 2114: Artificial Intelligence (3 CU)

(a) Description

This course examines the concepts, techniques, applications, and theories of Artificial Intelligence. The focus of the course is on the theory and application of artificial intelligence. Topics include logic, search, and reasoning with an emphasis on fundamentals and recent advances in AI. 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.

(b) Aims and Objectives

- To give students an advanced understanding of, and competence with, the theories, concepts, technologies and techniques of Computer Games development.
- To produce graduates possessing awareness, knowledge and practical skills in the field of Computer Games enabling them to follow a programme of study that will offer relevant specialization and career options.
- To develop students professional attitudes, interpersonal and entrepreneurial skills which are required by a practitioner in the industry.
- To provide students with critical and evaluative perspectives related to Computer Games development and develop student's capacity for independent and self-reflective learning, ensuring their future contribution to research and development.

(c) Learning outcomes

At the end of this course, the student will be able to:

- Formulate and assess problems in artificial intelligence.
- Assess the strengths and weaknesses of several methods for representing knowledge.
- Assess the strengths and weaknesses of several AI algorithms in areas such as heuristic search, game search, logical inference, statistical inference, decision theory, planning, machine learning, neural networks, and natural language processing.
- Implement software solutions to a wide-variety of problems generally considered to require artificial intelligence.

(d) Intellectual, Practical and Transferable skills

This course describes and discusses several algorithms and techniques within the fields of artificial intelligence and machine learning. These are algorithms and techniques that have practical applicability in computer science fields. Theoretical and practical understanding of these areas equips the student with the insights and tools required for solving complex and difficult problems, and for implementing them in software.

(e) Teaching and Learning Pattern

Teaching and learning is implemented through lecture, lab and tutorial sessions. Students are expected to make presentations of their assignments for discussion in class.

(f) Indicative Content

- Introduction to Artificial Intelligence: Simulation of Intelligence behavior, in different areas;
- Problem solving: games, natural language question answering, visual perception, learning; Aim-oriented (heuristic) algorithms versus solution-guaranteed algorithms.
- Understanding Natural Languages: Parsing techniques, context-free and transformational grammars, transition nets, augmented transition nets, grammar-free analyzers, sentence generation.
- Knowledge Representation: First-order predicate calculus; PROLOG and LISP languages; Semantic nets; partitioned nets; Production rules; knowledge base, the inference system, forward and backward deduction.
- Expert System: Existing systems (DENDRAL, MYCIN), Domain exploration; Meta-knowledge, expertise transfer, self-explaining systems.
- Pattern Recognition Structured Descriptions: Symbolic description, machine perception, line finding, interpretation, semantics and models, object identification and speech recognition.

(g) Assessment method

Assessment will be by continuous assessment through practical exercises and Coursework (40%) and final exam (60%)

(h) Reading List

(i) R. Duda, P.Hart, Pattern Classification and Scene Analysis, Wiley, 1973.

(ii) E.A. Feigenbaum, J.Feldman; Computers and Thoughts, AAAI Press, 1995.

(iii) N.J. Jilsson, Problem Solving Methods in Artificial Intelligence, McGraw- Hill, 1971.

(iv) J.Lloyd, Foundation of Logic Programming, Springer-Verlag, 1993.

(v) Artificial Intelligence: A modern approach 3rd Edition by Stuart Russell and Peter Norvig, 2009

6.5 Year 2 Semester 2

6.5.1 CSC 2200: Operating Systems (4 CU)

(a) Description

Operating Systems course introduces students to software that controls hardware and makes the hardware usable. Its interaction with other computer devises and how it controls other computer processes is explored.

(b) Aims

The aims of the course are:

- To provide students with a detailed understanding of how operating systems work.
- To provide students with skills to write basic programs to utilize underlying operating system infrastructures.

(c) Learning outcomes

At the end of this course, the student will be able to:

- Describe the dominant categories of operating systems
- Describe the difference between the dominant operating systems
- Describe the different design principles for operating systems and various software tools that make operating systems usable.

(d) Teaching and Learning Pattern

The teaching pattern is by lectures, lab sessions and group projects.

(e) Indicative Content

- Operating Systems Structures
- Processes and threads
- Thread creation, manipulation and synchronization
- Deadlock
- Implementing Synchronization operations
- CPU scheduling
- Memory management
- File systems and file system implementation
- Monitors
- Segments
- Disk Scheduling
- Networking
- UDP and TCP

(f) Assessment method

The assessment will constitute Practical assignments on at least 5 chapters of the course and written course work (40%) and written Exam (60%)

(g) Reading List

(i) Operating Systems: Internals and Design Principles 5th Ed by William Stallings, Prentice Hall, 2005.
(ii) Operating System Concepts by Abraham Silerschatz, Peter Baer Galvin; Addison-Wesley
(iii) Modern Operating Systems 3rd Edition by Andrew S. Tanenbaum
(iv) Lecture notes on Operating Systems by Marvin Solomon, University of Wisconsin Madison 2007
(v) Operating Systems and Middleware: Supporting controlled Interaction by Max Hailperin , 2011

6.5.2 CSC 1209: Logic Programming (3 CU)

(a) Description

This course introduces a paradigm where computation arises from proof search in a logic according to a fixed, predictable strategy. It thereby unifies logical specification and implementation in a way that is quite different from functional or imperative programming. This course provides a thorough, modern introduction to logic programming. It introduces the basic concepts and techniques of logic programming followed by successive refinement towards more efficient implementations or extensions to richer logical concepts. It covers a variety of logics and operational interpretations.

(b) Aims

The aim of the course is to provide a basic introduction to the logic programming language, Prolog. It aims at introducing a number of logical systems of importance in computer science.

(c) Learning outcomes

By the end of the subject, students should:

- Be conversant with the syntax and semantics of propositional and predicate logic
- Be familiar with a variety of applications of predicate logic in software verification, databases and knowledge-based systems
- Be able to write specifications in predicate logic expressing state constraints
- Understand the notion of formal proof, and be able to construct simple proofs in a natural deduction proof system for predicate logic
- Be aware that there are inherent expressiveness and computational limitations to the applicability of logical systems, and be familiar with a number of restrictions under which the computational limitations can be overcome
- Be able to write programs in a logic programming language,
- Understand both the top-down and the bottom up operational semantics of logic programs
- Be familiar with a logic for reasoning about sequential programs, and capable of constructing correctness proofs for simple programs

(d) Teaching and Learning pattern

The course consists of a traditional lecture component and a project component. The lecture component introduces the basic concepts and techniques of logic programming. The project component will be one or several projects related to logic programming.

(e) Indicative content

- Introduction
- Pure logic (relational) programming
- The Prolog Language
- Programming in Prolog.
- Efficient Prolog Programming
- Combining Logic Programming, Functional Programming, Higher Order, Objects.
- Review of first order predicate logic and resolution.
- Fundamental results.
- Semantics of logic programs.
- Implementation of logic languages and advanced compilation.
- Parallelism, concurrency.
- Other LP/CLP languages

(**f) Assessment method**

Assessment will be by assignments and/or tests (40%) and written examination (60%)

(g) Reading List

(i) Logic in Computer Science, Modeling and Reasoning about Systems, M.R. Huth and M.D. Ryan, Cambridge University Press 2000.
(ii) SWI Prolog Home Page, http://www.swi-prolog.org/
(iii) Logic Programming with Prolog by Max Bramer, 2005
(iv) A grammatical view of Logic programming by Pierre Deransart and Jan Maluszynski
(v) Logic Programming: Proceedings of the 1999 International Conference on Logic programming

6.5.3 CSC 2209: Systems Programming (4 CU)

(a) Description

Systems programming is aimed at teaching students how to write programs using system level services. The system of instruction is UNIX due to availability of free system tools that have been largely developed by and for the academia

(b) Aim

Skills in tools are provided by systems, their commands, system calls and understanding for model of computation.

(c) Learning outcomes

By the end of the subject, students should be able to:

- Design programs using system level services
- Describe the Standard I/O UNIX Library
- Describe the Files and Directories
- Describe interprocess communication

(d) Teaching and Learning Pattern

The teaching pattern is by lectures, lab sessions and projects.

(e) Indicative Content

- Introduction and Unix Standardization
- File input and output
- Standard I/O Library
- Files and Directories
- System Data Files and Information
- Process Environment
- Process Control
- Process Relationships
- Signals
- Threads
- Advanced I/O
- Interprocess Communication

(f) Assessment method

Assessment will be in form of tests and practical assignment (40%) and final written examination (60%)

(g) Reading List

(i) Advanced programming in the Unix Environment, by W. Richard Stevens, Addison-Wesley 2008
(ii) Systems Programming and Operating Systems by Dhamdhere ; McGraw Hill, 1999 ISBN 0074635794, 9780074635797
(iii) The Linux Programming Interface by Michael Kerrisk ; ISBN-10: 1593272200 , 2010
(iv) Advanced Programming in the UNIX Environment, 2nd Edition by Stephen A. Rago, Addison-Wesley

(v) UNIX Systems Programming: Communication, Concurrency and Threads by Kay A. Robbins

6.5.4 CSC 2210: Automata, Complexity and Computability (3 CU)

(a) Description

The course introduces students to the concept of automata and complexity. It sets a background for more advanced studies like compiler construction and principles of programming languages.

(b) Aims

The aims of the course are:

- To introduce students to the concepts of complexity, automata and computability
- To prepare students for advanced studies in compiler construction and principle of programming languages

(c)Learning outcomes

By the end of the subject, students should be able to:

- Discuss the concepts of complexity, automat and computability
- Apply the knowledge in compiler construction
- Explain finite state machines and regular languages
- Describe normal forms of grammar

(d)Teaching and learning pattern

Teaching will be in form of class lectures and tutorials

(e) Indicative content

- Finite state machines and regular languages:
- Deterministic and non - deterministic machines;
- Equivalence and minimization;
- Regular expressions and regular grammars;
- Kleene's theorem.
- Push-down automata and context free grammars;
- Normal forms of grammars; Top-down and bottom-up parsing.
- Turing machines and computability;
- Church's thesis;
- NP-Computable problems.

(f) Assessment method

Assessment will be in terms of Assignments and tests (40%) and final written examination (60%)

(g) Reading List

(i) John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman: Introduction to Automata Theory, Languages, and Computation; Addison Wesley, 2000.

(ii) Du, D.-Z. and Ko, K.-I. *Theory of Computational Complexity.* New York; Wiley, 2000

(iii) Introduction to Computer Theory 2/E Daniel I. A. Cohen John Wiley & Sons, Inc 1997.

(iv) Brookshear, J. G. *Theory of Computation: Formal Languages, Automata, and Complexity.* Redwood City, CA: Benjamin/Cummings, 1989.

(v) Introduction To Automata Theory, Languages and Computation by Hopcroft, Motwani, and Ullman

6.5.5 BIT 2207: Research Methodology (3 CU)

(a) Description

The purpose of this course is to acquaint students with types of scientific research relevant for anyone working in the field of computer science. It will enable students to develop capacity to conduct small, simple research projects while at the university.

(b) Aims

The aims of this course unit are to:

- Enable students become competent in understanding the research process;
- Provide skills that will enable students undertake independent research using a variety of appropriate methods, using primary and secondary data, as well as qualitative and quantitative techniques;
- Provide students with skills to produce a research proposal;
- Highlight ethical research practices to students.

(c) Learning outcomes

By the end of the course, students will be:

- Capable in their chosen professional, vocational or study areas to conduct research;
- Able to contribute in an entrepreneurial and innovative way within their business, workplace or community in the field of research;
- Able to operate effectively and ethically in conducting research in groups/teams
- Adaptable and manage change to handle different research situations according to different contexts;
- Aware of research and research methodology in subsequent years of study.

(d) Intellectual, Practical and Transferable skills

At the end of the course, students should have the ability to demonstrate:

- Appreciation of the different functions and applications of scientific research in the field of computer science;
- Basic knowledge of the different research methodologies relevant for computer science;
- Knowledge of which methods to use in what circumstances;
- Knowledge of what a research proposal entails;
- Application of quantitative and qualitative research methods and techniques;
- Judgment of the quality of research proposals as well as the products (articles, papers, theses etc.) of scientific research.

(e) Teaching and Learning Pattern

Teaching will be in form of formal lectures, tutorials and seminars. Classes will be interactive and students are expected to come to class prepared to participate and contribute regularly to class activities and discussions.

(f) Indicative Content

The content of this course will include:

- Introduction to scientific research;
- Formulating and clarifying the research topic and research problem;
- Conducting a literature review;
- Different research approaches;
- Ethics in research;
- Sampling;
- Use of secondary data;
- Collection methods for primary data;
- Analyzing qualitative data;
- Analyzing quantitative data and writing a research proposal and project report.

(g) Assessment method

The course will be assessed by course work and tests (40%) and final examination (60%)

(h) Reading List

(i) Cooper, H. (1998). Synthesizing Research: A Guide for Literature Reviews. Thousand Oaks, California: Sage Publications.

(ii) Saunders, M, Lewis, P & Thornhill, A (2003), Research Methods for Students, 3rd edn, UK, Financial Times, Prentice Hall.

6.6 Year 2 Recess Term

6.6.1 CSC 2303: Field Attachment (5 CU)

Description:

During Industrial training, students are to go and work in an organization with an IT department. The student is to be under the supervision of one of the workers in the organization. The student is assigned duties in line with the operations of the organization. Staff from Makerere University will make visits to get the students' view of the organization as well as the organization's view about the student. The supervisor will be given a form to evaluate the students and the student will make a report about his experience. The two reports will be used to evaluate the student.

6.7 Year 3 Semester 1

6.7.1 CSC 3110: User Interface Design (4 CU)

(a) Description:

The course introduces the principles of user interface development, focusing on design, implementation and evaluation.

(b) Aims

The course aims at providing the skills listed below to students:

- Developing efficient, flexible and interactive User Interfaces(UI)
- Provide ability to identifying system users, the tasks they want to carry out and the environment in which they will be working;
- Creating conceptual designs;
    - Designing various kinds of UI, in particular graphical user interfaces (GUIs) , computer and mobile and web sites;
- Evaluating UIs
- Appreciation of realities of developing usable UIs in an organization.

(c)Learning outcomes

By the end of the subject, students should be able to:

- Design efficient, flexible and interactive user Interfaces
- Demonstrate ability identifying systems users and the tasks they want to carry out
- Evaluate user interfaces
- Appreciate realities of developing usable UIs in an organization

(d) Teaching and Learning Pattern

The teaching pattern is by lectures, lab sessions and projects.

(e) Indicative content

- Usability
- User-Centered Design
- UI Software Architecture
- Human Capabilities
- Output Models
- Conceptual Models and Metaphors
- Input Models
- Design Principles
- Paper Prototyping
- Constraints and Layouts
- Graphic Design
- Experiment Design
- Computer Prototyping
- Heuristic Evaluation
- User Testing
- Experiment Analysis

(f) Assessment method

Assessment will be in form of assignments and tests (40%), practical Exam (30%) and final written exam (30%)

(g) Reading List

(i) Norman, D. A. The Design of Everyday Things. New York, NY: Doubleday, 1990. ISBN: 0385267746.

(ii) Nielsen, J. Usability Engineering. Burlington, MA: Academic Press, 1994. ISBN: 0125184069.

(iii) Mullet, K., and D. Sano.Designing Visual Interfaces: Communication oriented techniques. Prentice Hall, 1994. ISBN: 0133033899.

6.7.2 BAM 2102: Entrepreneurship Principles (3 CU)

(a) Description:

The course introduces the students to the basic concepts in entrepreneurship, identification of business opportunities, business evaluation and analysis. It provides students with the skills needed to effectively identify, organize, develop, and manage own business ventures. This course is based on creativity and professional development foundations that should orient a student to take adventure, a personal journey, and a seize opportunity for business start-up. The course gives students an opportunity to make creative adjustments to meet personal needs and increase self-drive to achieving success.

(b) Aims:

A student that undertakes this course should be able to:

- Understand the origins of entrepreneurship and an entrepreneur
- Identify, evaluate, and select business opportunities
- Perform a self-evaluation to match their own characteristics with that of an entrepreneur
- Carry out feasibility and viability of an investment opportunity
- Analyze and exploit the Entrepreneurial Environment provided by the political, socioeconomic and technological conditions.

(c) Learning Outcomes:

On completion of this course unit, the students will be able to:

- Perform self-evaluation to match business opportunities

- Analyze the entrepreneurial environment
- Ensure start-up, survival, sustainability of an investment opportunity, identify their own personal
- entrepreneurial potential, ability, and competences
- Identify, and exploit business opportunities and resources

(d) Teaching and Learning pattern:

The teaching and learning approaches will combine classroom lectures, discussions and group activities, quizzes and take home assignments. A group project shall form part of the coursework. The material presented in class will overlap that of the text but will contain additions and variations.

(e) Indicative content:

- Entrepreneurship-Scope, theories;
- Entrepreneurial Process; the Entrepreneur, Creativity and Innovation, feasibility study and analysis, business planning,
- Creating and developing a business, Entrepreneurship, Entrepreneurship Development,
- Role of government in entrepreneurship growth, development and entrepreneurship

(f) Assessment method:

Assessment will be in terms of tests and practical exercises (40 %) and a final examination (60%)

(**g) Reading List**

(i) Bruce R. Barringer & R. Duane Ireland (2006). Entrepreneurship: Successfully Launching New ventures. Published by Pearson-Prentice Hall. 1/e Edition. ISBN 0-13-061855-1

(ii) Thomas W. Zimmerer and Norman M. Scarborough (2005) Essentials of Entrepreneurship and Small Business Management. 4th Ed. ISBN 0-13-191856-7

6.7.3 CSC 3115: Advanced Programming (3 CU)

(a) Description

This course highlights programming practices that are vital in the day today work of a programming Professional. While many systems are described by functionalities, some important aspects like security, robustness, and maintainability are ignored. Students are to get an in depth understanding of these concepts as well as exploring the current trends in the programming environment.

(b) Aims

The aim of the course is to concretize the student's past programming experience as well as highlighting critical practices in programming that are necessary for a professional programmer

(c) Learning outcomes

By the end of the course, the student should be:

- Able to implement non- functional but critical aspects of programming like robustness and security
- Able to develop well documented and well- structured software that can easily be maintained
- Knowledgeable in other programming practices like mobile programming
- Aware of newer programming paradigms like service oriented and cloud computing

(e) Teaching and Learning Pattern

Teaching will be by lectures and lab demonstrations.

(f) Indicative Content

- Programming for Security:
- Programming for Robustness
- Programming for Maintainability
- Trends in Programming Paradigms

(g) Assessment method

Assessment will be by tests and practical assignments (40%) and final written examination (60%)

(h) Reading List

(i) Advanced Programming in the UNIX Environment by W. Richard Stevens and Stephen A. Rago Addison Wesley 1992
(ii) Test-Driven Development: A Practical Guide, by David Astels
(iii) Beautiful Code: Leading Programmers Explain How They Think, by Andy Oram & Greg Wilson
(iv) Refactoring: Improving the Design of Existing Code, by Martin Fowler
(v) Refactoring to Patterns, Joshua Kerievsky

6.7.4 CSC 3112: Principles of Programming Languages (3 CU)

(a) Description

The course introduces students to the low level organization and operation of programming languages. It covers semantic and syntactic as well as operational issues in programming languages. The building blocks of programming languages are explored.

(b) Aims

The aims of the course are

- To give students fundamental knowledge in the organization and operation of programming languages
- To make students appreciate the possible future evolutions of programming languages
- To expose students to causes of operational (like performance, security, etc) characteristics of programming languages

(c) Learning outcomes

By the end of the course, students will be able to:

- Understand common language paradigms
- Know the different building blocks of a programming language
- Know how the different blocks of a programming language interact

(d) Teaching and Learning Pattern

Teaching will be largely by lectures, tutorials and class assignments

(e) Indicative Content

- Overview over programming language paradigms
- Common principles: syntax, syntax trees, formal semantics (denotational and operational), variables and binding
- Types: role of types in programming and programming languages, types and their operations: products, sums, functions, recursive types, reference and array types
- Primarily imperative issues: control flow, arrays, pointers and references, parameter-passing mechanisms, scoping
- Type systems: strongly typed languages type checking (static vs. dynamic), type equivalence (by name vs. structural), overloading, coercion, polymorphism, type inference
- Binding: declarations and environments. Block structure: scope and visibility, stack discipline. Bound occurrences: static vs. dynamic binding.
- Encapsulation: information hiding, modules, abstract data types, classes
- Language implementation: parsing, code generation, garbage collection.

(f) Assessment method

Assessment will be by Tests and Assignments (40%) and final written examination (60%)

(g) Reading List

(i) Friedman,Wand, and Haynes, Essentials of Programming Languages, MIT Press, 2001
Manual (5*th* Edition) by Samuel P. Harbison; Prentice Hall, 2002.
(ii) Principles of Programming Languages by Gilles Dowek , Springer 2009
(iii) Principles of Programming Languages by Pradnya Kashikar, Nilesh M. Magar
(iv) Principles of Programming Languages: Design, Evaluation, and Implementation by Bruce J. MacLennan
(v) Principles of Programming Languages version 0.7 by Mike Grant, Zachary Palmer and Scott Smith

6.7.5 BIS 3100: Modeling and Simulation (3 CU)

(a) Description

The course gives students theoretical and practical skills in modeling and simulation of dynamic systems with a view of learning their behavior and the sensitivity of that behavior to certain parameters.

(b) Aims

The aims of the course are:

- Familiarize students with modeling and simulation techniques that are applicable under varying circumstances
- Equip students with practical experiences of composing models and running simulations under varying circumstances
- Equip students with skills of correctly representing simulation results

(c)Learning outcomes

By the end of the subject, students should be able to:

- Describe modeling and simulation techniques
- Apply experience of composing models and running simulations under varying circumstances

(d) Teaching and Learning pattern

Teaching and Learning will be in form of Lectures and laboratory demonstrations

(e) Indicative Content

- Simulation of operational systems
- Simulation as a decision making methodology
- Model development and validation,
- Design of simulation experiments,
- Generation of appropriate values of random variables,
- Interactive procedures and interpretation of results.

(f) Assessment Method

Assessment will be in form of tests and assignments (40%) and final examination (60%)

(g) Reading List

(i) Business Modeling and Simulation by Les Oakshott, 1997, Trans- Atlantic Publications
(ii) System dynamics modeling a practical approach by R.G. Coyle, Chapman & Hall/CRC, 1996.

6.7.6 CSC 3121: Computer Graphics (3 CU)

(a) Description

The course covers general purpose graphics systems and their use. It gives an in depth knowledge of computer graphics and graphical user interfaces.

(b) Aims

The aims of the course are:

- Introduce students to the concepts of graphical representation on computers
- Teach students the design of good graphical user interfaces

(c)Learning outcomes

By the end of the subject, students should be able to:

- Describe the concepts of graphical representation on computers
- Apply graphics systems and concepts to solving real life challenges
- Design good graphical user interfaces

(d) Teaching and Learning Pattern

Teaching will be in terms of class lectures and tutorials

(e) Indicative Content

- Graphics hardware,
- Geometrical transformations,
- Surface and volume visualization,
- Design and implementation of graphical user interfaces.
- Two dimensional imaging processes.
- Computer graphics applications.
- Display system organization;
- Display devices and modes;
- Display file construction and its structure;
- Graphic primitive - device initialization,
- View porting and windowing;
- Line drawing,
- Simple and symmetrical Digital Differential Analysis (DDA);
- Arch and circle generating DDA Line; and polygon clipping algorithms;
- Curve plotting;
- Transformations- projections and perspective views;
- Picture segmentation: Graphics standards - PHIGS and GKS.

(f) Assessment method

Assessment will be in terms of assignments and tests (40%) and final exam (60%)

(g) Reading List

(i) Introduction to Computer Graphics by James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes and Richard L. Phillips, Addison Wesley, 2003.

(ii) Fundamentals of Computer Graphics by Peter Shirley. AK Peters, 2002.

6.7.7 CSC 3118: Computer Science Project I (5 CU)

(a) Description

The course is to allow students, individually or in groups, to integrate the knowledge acquired over the previous five semesters into solving a non - trivial problem through a computer application. Emphasis will be put on the systematic development methodology, the documentation of the development process. The expected output at this stage will be the concept paper for the third year project in the semester 6.

(b) Aims

The aim of the course is to give students experience in

- Individual and collaborative work
- Proper procedures in development of computer systems
- Proper documentation of the software development process

(c)Learning outcomes

By the end of the subject, students should be able to:

- Demonstrate skills acquired in the entire course
- Apply the skills and knowledge acquired during the course of study to come up with an acceptable proposal

(d) Teaching and Learning pattern

Students will be under the supervision of a member of staff (at least at a rank of Assistant Lecturer). The supervisor will guide them in the day today progress of the project. When the supervisor feels the students have addressed the problem at hand, (s)he will sign off their concept paper and recommend them to continue with the implementation.

(e) Indicative content

The content is to be determined by the students under the guidance of the supervisor.

(f) Assessment Method

Students will submit the signed Proposal to the department. The department will appoint a supervisor who will work with the students as they come up with their Proposal. Students will be required to present their work and answer any questions from the supervisor.

6.7.8 MTH 3107: Linear Programming (3 CU)

(a) Description

The course is to introduce students to the broad concepts of Linear Programming. Students will learn how to interprete and analyze LP problems, formulate them as problems and use existing techniques to solve them.

(b) Aims

The aim of the course is to improve students' problem solving skills by subjecting them to real life problems and guide them through formulation of their solutions. The choice of the cases chosen depends on their applicability in real life computing environment.

(c) Learning outcomes

By the end of the course, the student should be able to

- Correctly formalize real life problems into OR problems
- Adequately solve typical OR problems
- Make post optimality analysis on OR solutions

(d) Teaching and Learning Pattern

Teaching will be in form of lectures and Tutorials

(e) Indicative Content

- General LP Problem
- Algebra and geometry of LP problems
- Geometric solutions to LP problems, basic and optimal solutions to an LP problem
- The constraint set as a convex polytype
- The connection between extreme points and the basic solutions
- The simplex method
- The big M method
- The Dual simplex method
- The mutual primal-dual simplex algorithm
- Post optimality analysis

(g) Assessment method

Assessment will be by assignments and tests (40%) and final written exam (60%)

(h) Reading List

(i) Linear Programming and Network Flows by Mokhtar S. Bazaraa, John J Jarvis and Hanif D Sherali, John Wiley, 2005
(ii) Operations Research: Applications and Algorithms by Wayne L. Winston. Wadsworth Publishing Company, 1997.
(iii) Vasek Chvatal, Linear Programming
(iv) Dr. Saul I. Gass , Linear Programming: Methods and Applications 5th Edition
(v) Stephen Boyd and Lieven Vandenberghe, Convex Optimization, 2004

6.8 Year 3 Semester 2

6.8.1 BSE 2206: Data Communications (4 CU)

Pre-requisites: Computer Networks

Course Description

This is a theoretical course that covers the fundamentals of data communication Formatting and transmission of digital information over various media

Aims

The aims of the course are

- To provide a solid basis on the theoretical and practical understand of data communication networks
- To introduce students to standards and guidelines in computer and data communication networks
- To impact knowledge and skill relevant for the design, implementation and maintenance of modern computer communication networks
- To introduce students to emerging technologies in data communication

Learning outcomes

Upon successful completion of this course, the student should be able to

- Describe theoretical concepts of computer communication systems, including standards, protocols, and infrastructure devices e.g. routers and switches.
- Design, install and manage a simple Local Area Network or A campus Wide Area Network
- Describe emerging communication technologies particularly wireless and fiber optic technologies
- Describe emerging research directions in computer communication networks

Teaching & Learning patterns

This course will be delivered through class lectures, discussions. Simulations will also be used to explain abstract concepts.

Indicative content

- Introduction to communication
- Digital versus Analog transmission; Modems
- Transmission media: magnetic media, twisted pair, coaxial, fiber-optics;
- Data encoding: straight, Manchester, differential Manchester, satellite;,
- Modulation and their standards, codes and pulse code modulation;
- Integrated Services Digital Networks
- (ISDN);
- Network Access Protocols; Passive versus dynamic allocation;
- LAN standards:802.3 (Ethernet),802.4 (token bus), 802.5 (token ring);

Assessment

(Assignments, Tests, Group Research course work) 40%
Final Examination 60%

Reading List

(i) Data Communications and Networking, Behrouz A. Forouzan: Fourth Edition

6.8.2 CSC 3205: Compiler Design (3 CU)

(a) Description

In this course unit, students shall understand the complete process of translating a program in a high-level language to machine language. 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.

(b) Aims

The aim of the course is to allow students to examine how a high-level language program is accepted as input and translated into assembly language or machine language so that the central processing unit receives instructions which it understands and can execute.

(c)Learning outcomes

By the end of the subject, students should be able to:

- Describe the complete process of translating a program in a high level language to machine language
- Describe the design and implementation of a compiler
- Describe the principles and techniques for program analysis and translation
- Identify the tools for compiler construction

(d) Teaching and Learning pattern

The course consists of a traditional theoretical component and a project component. The lecture component introduces the basic concepts of compiler writing. The project component will involve students in writing a compiler for a specified programming language.

(e) Indicative content

- Language translators: Introduction to compilers and interpreters.
- The structure of a compiler: lexical analysis, parsing, semantic analysis.
- Intermediate code generation, register allocation, global optimization.
- Lexical scanning
- Parsing
- Automatic parser construction. FIRST and FOLLOW functions. LL(1) parsers. LR parsers. Conflicts in LR grammars and how to resolve them
- Semantic analysis: Attributes and their computation, tree-traversals, visibility and name resolution. Inherited attributes and symbol tables. Name resolution in block-structured languages
- Type checking: Type systems, varieties of strong typing, overload resolution, polymorphism and dynamic dispatching. Type-checking and type inference, unification
- Run-time or Run-time organization: storage allocation, non-local references, parameter passing, dynamic storage allocation. Exception handling, debugging information
- Intermediate code generation: control structures, expressions, simple register allocation. Aggregates and other high-level constructs
- Global optimization

(f) Assessment method

Assessment will be by assignments and/or tests (40%) and written examination (60%)

(g) Reading List

(i) Compiler construction by William McCastline Waite, Gerhard Goos Springer Verlag 1994
(ii) Alfred V. Aho , Monica S. Lam and Jeffrey D.Ullman Compilers: Principles, Techniques, and Tools (2nd Edition) 2000
(iii) Y.N. Srikant and Priti Shankar, The Compiler Design Handbook: Optimizations and Machine Code Generation, Second Edition, 2007
(iv) Kenneth C. Louden, Compiler Construction: Principles and Practice 1997
(v) Steven Muchnick , Advanced Compiler Design and Implementation 1997

6.8.3 CSC 3211: Computer Science Project II (5 CU)

(a) Description

The course is to allow students, in groups, to integrate the knowledge acquired over the previous five semesters into solving a non - trivial problem through a computer application. Emphasis will be put on the systematic development methodology, the documentation of the development process, and how well the developed systems address the problem to be solved. Non - functional attributes like robustness, usability, security and reliability will also be tested.

(b) Aims

The aim of the course is to give students experience in

- Group and collaborative work
- Proper procedures in development of computer systems
- Proper documentation of the software development process
- Development of big not trivial computer projects.

(c)Learning outcomes

By the end of the subject, students should be able to:

- Demonstrate skills acquired in the entire course
- Apply the skills and knowledge acquired during the course of study to come up with a working or running prototype or system

(d) Teaching and Learning pattern

Students will be under the supervision of a member of staff (at least at a rank of Assistant lecturer). The supervisor will guide them in the day today progress of the project. When the supervisor feels the students have addressed the problem at hand, (s)he will sign off their report and recommend them for examination.

(e) Indicative content

The content is to be determined by the students under the guidance of the supervisor.

(f) Assessment Method

Students will submit the signed report for examination to the department. The department will appoint an examination panel of at least 5 people who will evaluate the report as well as testing the system. Students will be required to present their work and answer any questions from the panel. Each member of the panel will award a mark depending on his/her view on the worthiness of the developed application.

6.8.4 CSC 3207: Computer Security (3 CU)

(a) Description

Computer security is a branch of technology concerned with digital security or information security applied to computers. Since the largest part of the computer that users interact with is software, computer security pays big attention to development of secure software.

(b) Aims

The aims of the course are:

- To introduce students to threats faced by computers in the connected digital world.
- To introduce students to techniques that are used to protect computers against various threats.

(c)Learning outcomes

By the end of the subject, students should be able to:

- Discuss the threats faced by computers in the digital world
- Describe the techniques that are used to protect computers against various threats
- Describe digital security principles
- Describe Access control lists

(d) Teaching and Learning Patterns

The teaching pattern is by lectures, lab sessions and group projects

(e) Indicative content

- Digital security principles
- Hardware based security mechanisms
- Secure operating systems
- Security architecture
- Security by design

- Secure coding (Software Security)
- Access Control Lists
- Security Applications

(f) Assessment method

Assessment will constitute Practical assignments on at least 5 chapters of the course and written course work (20%) and written Exam (60%).

(g) Reading List

(i) Ross J. Anderson: Security Engineering: A Guide to Building Dependable Distributed Systems, Willey 2001.
(ii) Robert C. Seacord: Secure Coding in C and C++. Addison Wesley, 2005.
(iii) Keith M. Jackson, Jan Hruska, Donn B. Parker, Computer Security Reference Book, 1992
(iv) Bruce Schneier, Fast Software Encryption
(v) Katie Hafner, Cyber Punk

6.8.5 BIS 3205: Data Warehousing and Business Intelligence (4 CU)

(a) Description

This course covers techniques and software tools that can assist management that deals with large amounts of data in management and business decision making. The course covers the fundamental differences between databases and data warehouses, the techniques of developing data warehouses as well as manipulating them to generate business strategic decisions.

(b) Aims

The aims of the course are

- To give students the understanding on the role and operation of data warehouses
- To equip students with skills of developing data warehouses
- To equip students with skills of maintaining existing data warehouses
- To equip students with skills of manipulating data warehouses to generate information for business decision making

(c) Learning outcomes

By the end of the course, the student should be able to

- Distinguish the roles of a database from that of a data warehouse
- Develop a data warehouse
- Populate and manipulate a data warehouse

(d) Teaching and Learning Pattern

Teaching will be in form of class lectures, tutorials, lab demonstrations as well as class presentations.

(e) Indicative Content

- Data warehouse concepts: partitioning, granularity, record of source, and meta data
- Building viable decision support environments.
- Architect development,
- Data migration and integration,
- Use of operational data stores, and transactional systems.

(f) Assessment method

Assessment will be in form of tests and practical assignments (40%) and final written examination (60%)

(g) Reading List

(i) Data warehousing fundamentals by Paulraj Ponniah Wiley 2001.

6.8.6 BSE 3202: Distributed Systems Development (4 CU)

(a) Description

This course gives students theoretical and practical skills on development of distributed systems and applications. This includes distributed-system specific challenges like reliability and robustness.

(b) Aims

The aim of the course is to equip students with skills of developing distributed systems

(c) Learning outcomes

By the end of the course, the student should be able to

- Describe Event driven Architectures
- Describe the development, documentation and testing of distributed applications
- Describe the techniques of reusable, extensible and efficient software systems
- Discuss maintainability and concurrence in distributed systems
- Describe abstraction based on patterns and object-oriented techniques

(d) Teaching and Learning patterns

Teaching will be by class lectures and laboratory demonstrations

(e) Indicative content

- Event-driven software architectures,
- Distributed object computing,
- Development, documentation and testing of distributed applications
- Techniques for reusable, extensible and efficient software systems
- Maintainability and concurrence in distributed systems
- Abstraction based on patterns and object-oriented techniques

6.8.7 CSC 3217: Emerging Trends in Computer Science (3 CU)

(a) Description

The course is to expose provide students with an opportunity to search for knowledge in an area of interest. It is to allow a student do lightweight research and explore the current trends in a certain computer science area.

(b) Aims

The aims of the course are:

- To aid a student get an in depth understanding of the developments in one area of computer science
- To improve the student's research skills
- To develop confidence in the students on the ability to search for knowledge with little guidance.

(c)Learning outcomes

By the end of the subject, students should be able to:

- Discuss the various areas in computer science that are potential areas of research
- Demonstrate skills acquired in research methods

(d) Teaching and Learning pattern

Students will be grouped in theme areas which will be covered by a set of staff. Staff will guide students on the specific themes/topics in the area as well as where to search for information. Staff will also address experiences and hardships found.

(e) Indicative content

The content is not specific but will be dependent on the area the student chooses to pursue.

(f) Assessment Method

Students will present their findings in a report. The study report will be an outline and explanation of what the student has found out in the state of practice in the area of choice. The write up will be evaluated and the final mark awarded.

(g) Reading List

(i) This will be decided by the lecturer in consultation with the Course leader

(f) Assessment Method

Assessment will be in form of tests and (practical) assignments (40%) and final examination (60%)

(g) Reading List

(i) S. Tanenbaum and M. V. Steen, Distributed Systems: Principles and Paradigms, Second Edition, Prentice Hall, 2006.

(ii) R. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, John Wiley & Sons, 2001.

7. Resources and Infrastructure

The Department of Computer Science and the School of Computing and Informatics Technology have the enough resources and infrastructure to sufficiently run the programme.

7.1 Funds

Fees payable by the students will enable the University to sustain the programme. The fees and the number of students to be admitted are the same as those on the old curriculum.

7.2 Staff

The School of Computing and Informatics Technology has a big pool of staff who can competently teach the courses. The list of staff members in the Department of Computer Science is in Appendix B. The department also relies on staff from other departments like Information Technology, Information Systems and Networks together with visiting staff from other universities.

7.3 Lecture Space

Initially, the School of Computing and Informatics Technology housed in a 2,500 square meter building (Block A). In January 2009, a new 12,000 square meter building (Block B) was officially opened. Block B has Six Lecture Theaters each of which can accommodate up to 500 students.

7.4 Computer Laboratories and Software

The College building (Block A and Block B) have general laboratories (strictly for students practice), teaching laboratories and specialized laboratories. These laboratories are shared among the departments of the School and are scheduled by the ICT services unit. In all, the School has 6 general laboratories and 4 specialized laboratories. Currently, the School has approximately 2000 computers and 5000 students.

On top of the physical computers, students need software for the different practical sessions. Different computers are installed with different software depending on their focus. Most of the software is available as free distributions for academic purposes. The School and department therefore have (and can access) enough software that can run the practical aspects of the program.

7.5 Library Services

Makerere University Library supports the College Computing and Information Science Library which is located on the First level (Block B Building). The College Library is stocked with up-to-date information resources. The information resources in the College Library have been acquired through purchases made by Makerere University Library and the Faculty. In addition to this facility, the University Library provides access to print books, print journals, electronic journal databases, a well-stocked reference section and connections to many remote databases like the Uganda Scholarly Digital Library at http://dspce3.mak.ac.ug. The print collection is beefed up by the broad variety of electronic resources provided by the University Library and accessible online at http://muklib.mak.ac.ug. Through the Document Delivery Service which is provided by the University Library, users who fail to get access to full-text articles from the available databases can make requests for the articles and delivered to them at no cost. Library users can also access the Online Public Access Catalogue (OPAC) to get bibliographic information about the collections found in the College Library at http://196.43.133.123:8080.

8. Quality Assurance

Several activities will be carried out as quality assurance measures so as to:

(a) Measure the general extent to which the required skills have been achieved

(b) Ascertain the implementation of the methodological changes proposed

(c) Create a feedback benchmark for possible future revisions in the curriculum

The following activities will be carried out in the process of monitoring and assuring quality in the proposed program.

8.1 Feedback from students enrolled

In the current set up, each class has 4 student representatives (2 day, 2 evening). These representatives are in constant contact with the Head of Department in case there are any quality related matters in a particular class. This set up is to be maintained. At the end of the semester, samples of students are given questionnaires to respond to several quality related matters like staff punctuality, delivery mode, course content and the general perceived usefulness of the course unit.

The School of Computing and Informatics Technology has a computerized system that captures and analyzes the data. With the computerized system:
(i) Every student is required to assess every lecturer teaching him/her, the sample space will therefore be increased
(ii) No time is required in the analysis of the results. Staff and School management will be able to get the feedback instantly
(iii) Data is easily archived and therefore the trend of staff performance in specific areas will be easy to visualize

8.2 Class meetings

The departmental management makes at least one (1) meeting with every class every semester. In this meeting, general quality issues are addressed. Students are also given a chance to raise any questions that are answered and/or addressed by the department management. This set up will also continue

8.3 Use of ICT in availing lecture materials

Currently, Makerere University has MUELE an e-learning tool on its Intranet. Students in the Department of Computer Science have adequate access to computers. This creates a good environment for e-learning blended teaching. All courses in the revised curriculum will be taught in a blended way. All course materials will be put on MUELE. Staff will, as much as possible, make use of e-learning facilities like discussion forum and drop boxes for assignments. This will increase student activity/participation and reduce staff effort (e.g. staff will not need to dictate notes). This in turn will, in turn, increase the material covered and taken in by the students.

8.4 Peer review

Course leaders and other members of staff will enroll (as students) to all classes taught in the department. They will therefore be able to view contents of courses taught by their peers. Course leaders will advise fellow staff on the content, depth and presentation of materials. Consequently, for every course, students will access the best material provide on the online platform which is also viewed by all staff in the department. But the course instructor shall be excluded in this view.

8.5 External examiners' reports

Like it is everywhere in Makerere University, student results are reviewed every semester by a senior external academician. This is to bring a 'foreign view' of the quality of the program. External examiners write reports on their view of the curriculum/examinations. Some recommendations can be implemented immediately while others have to be implemented in a longer term. The department will make the maximum possible use of external examiners' reports as a means of assuring quality in the program.

8.6 Industrial Training placement reports

Students will get Industrial placements in the programme. This will help expose them to the world of the industry. Reports from the placements will be used to gauge the effectiveness of the curriculum. They are also to be used to identify and address the deficiencies in the curriculum.

8.7 Tracer studies

The School of Computing and Informatics Technology is devising ways of keeping in contact with its alumni together with their employers. This is with a view of making a tracer study of its graduates. The Department of Computer Science will use outputs of the tracer studies to gauge the quality of the program and whenever necessary, improve it.

9. Staff in the Department of Computer Science

No	Name (Highest Qualification	Qualification	Rank	Specialization
1	John Quinn	PhD	Lecturer	Computer Science
2	John Ngubiri	PhD	Senior Lecturer	Computer Science
3	Florence Tushabe	PhD	Senior Lecturer	Computer Science
4	Joseph Balikuddembe	PhD	Lecturer	Software Engineering
5	Engineer Bainomugisha	PhD	Lecturer	Software Engineering
6	Richard Ssekibuule	MSc	Ass. Lecturer	Computer Science
7	Jonathan Kizito	MSc	Ass. Lecturer	Software Engineering
8	Rose Nakibuule (M.Sc)	MSc	Ass. Lecturer	Computer Science
9	Doreen Tuheirwe (M.Sc)	MSc	Ass. Lecturer	Software Engineering
10	Swaib Dragule (M.Sc)	MSc	Ass. Lecturer	Computer Science
11	Marriette Katarahweire (M.Sc)	MSc	Ass. Lecturer	Software Engineering
12	Michael Kizito (M.Sc)	MSc	Ass. Lecturer	Computer Science
13	Paul Bakaki (M.Sc)	MSc	Ass. Lecturer	Computer Science
14	Diana Nakiyingi (M.Sc)	MSc	Ass. Lecturer	Computer Science
15	Barbara Nansamba (B.Sc)	BSc	Teaching Assistant	Computer Science
16	Emmanuel Lule (B.Sc)	BSc	Teaching Assistant	Computer Science
17	Joseph Derrick Olaka (B.Sc)	BSc	Teaching Assistant	Computer Science
18	Maureen Nyonyintono (B.Sc)	BSc	Teaching Assistant	Computer Science
19	Safari Yonasi (B.Sc)	BSc	Teaching Assistant	Computer Science