गणितीय विज्ञान संकाय
FACULTY OF MATHEMATICAL SCIENCES
दिल्ली विश्वविद्यालय, दिल्ली – 110007 (भारत)
UNIVERSITY OF DELHI, DELHI – 110 007 (INDIA)

Email: dean_mathsci@du.ac.in Ph.27666041

Prof. Naveen Kumar, Dean
प्रोफेसर नवीन कुमार, अधिष्ठाता

संदर्भ सं.: Math.Sci./2023/
तिथि:November 20, 2023

MINUTES

An emergent meeting of the Faculty of Mathematical Sciences was held on Monday, 20th November, 2023 at 02.30 p.m. in the Seminar Room, 1st Floor, New Academic Block, Faculty of Mathematical Sciences Building, University of Delhi, Delhi-110 007.

Sl.No.	Name	Department/ College
01.	Prof. Naveen Kumar Dean, Faculty of Mathematical Sciences & Head, Department of Computer Science	Chairperson
02.	Prof. Ruchi Das	Head, Mathematics
03.	Prof. Ranjita Pandey	Head, Statistics
04.	Prof. Preeti Wanti Srivastava	Head, Operational Research
05.	Sr. Prof. Poonam Singh	Statistics
06.	Prof. Arvind Patel	Mathematics
07.	Prof. Ompal Singh	Operational Research
08.	Dr. Om Pal	Computer Science
09.	Prof. Archana Singhal	I P College
10.	Dr. Neeraj Kumar Sharma Special Invitee	Ram Lal Anand College
11.	Dr. Nidhi Arora	Kalindi College
12.	Dr. Preeti Marwah	Acharya Narendra Dev College
13.	Dr. Bhawna Bansal Gupta	Miranda House
14.	Dr. Pooja	Kamla Nehru College

Prof. Riddhi Shah, Member Expert, Jawahar Lal Nehru University expressed her inability to attend the meeting:

It was resolved as under:

1. The Committee considered the recommendations of the Under-Graduate Committee of Courses & Studies for (meeting held on 20.11.2023) of the **Department of Computer Science.**

   It was resolved to accept the recommendation of the Committee constituted by the competent authority to consider the matter relating to the B.Sc. Program and recommend to the Academic Council that the following may be considered for notification to all concerned:

   B.Sc. with Computer Science as per the UGC guidelines is synonymous to:
   - **B.Sc. (Physical Science), and B.Sc. (Applied Physical Science) where Computer Science is a discipline.**
   - **B.Sc. (Mathematical Sciences), where Computer Science is a discipline.**

2. The committee reviewed the changes in the M.Tech Computer Science structure and recommended that the same may be approved by the Academic Council.

The meeting ended with a vote of thanks to the Chair.

DEAN & CHAIRPERSON

Learning Outcomes based Curriculum Framework (LOCF)
for
Computer Science
Undergraduate B.Sc./B.Sc.(Hons) Programmes
2020

UNIVERSITY GRANTS COMMISSION
BAHADUR SHAH ZAFAR MARG
NEW DELHI – 110 002

Table of Contents

Preamble

Education is the key to development of any society. Role of higher education is crucial for securing right kind of employment and also to pursue further studies in best available world class institutes elsewhere within and outside India. Quality education in general and higher education in particular deserves high priority to enable the young and future generation of students to acquire skill, training and knowledge in order to enhance their thinking, creativity, comprehension and application abilities and prepare them to compete, succeed and excel globally. Sustained initiatives are required to reform the present higher education system for improving and upgrading the academic resources and learning environments by raising the quality of teaching and standards of achievements in learning outcomes across all undergraduate programs in science, humanities, commerce and professional streams of higher education including computer science. One of the significant reforms in the undergraduate education is to introduce the Learning Outcomes-based Curriculum Framework (LOCF) which makes it student-centric, interactive and outcome-oriented with well-defined aims, objectives and goals to achieve. LOCF also aims at ensuring uniform education standard and content delivery across the country which will help the students to ensure similar quality of education irrespective of the institute and location. With initiatives of University Grants Commission (UGC) for nation-wide adoption and implementation of the LOCF for bachelor's programmes in colleges, universities and HEIs in general. A Core Expert Committee (CEC) was constituted to formulate the modalities for developing the LOCF in various subjects being taught in the undergraduate courses in sciences, humanities, commerce and professional courses. The CEC also constituted the Subject Expert Committees (SEC) in various subjects to prepare detailed guidelines for the LOCF in subjects concerned.

The Learning Outcomes (LO) specified by the CEC are the guidelines to determine the structure of the undergraduate programs to be offered by the Higher Educational Institutions (HEI) of our country. The key components of the planning and development of LOCF are given in terms of clear and unambiguous description of the Graduate Attributes (GA), Qualification Descriptors (QD), Program Learning Outcomes (PLO) and Course Learning Outcomes (CLO) to be achieved at the end of the successful completion of each undergraduate program to be offered by HEIs. In undergraduate education in Computer Science, there are two programmes of study leading to the degree of B.Sc. with Computer Science and B.Sc(Hons) in Computer Science. Several meetings were held by the SEC to formulate the framework for both undergraduate programmes. In the first meeting of the Committee, held in UGC on …, the Chairman of SEC briefed the

members about the decisions taken in the meeting of chairpersons of all SECs with the members of CEC and officers of UGC. He appraised the members the task at hand and the modalities to prepare the report were elucidated. The topics were allocated to each member keeping in mind the members' expertise and interests. It was proposed that the prepared notes shall be circulated among all members for feedback in the first instance. The committee after getting first set of inputs met again at University of Hyderabad on October 8, 2018 where different course outcomes, course objectives, learning outcomes, core structures of the programme were discussed. Chairman also informed that UGC also wants detailed syllabus of each course. Accordingly, each member was advised to prepare the syllabus along with textbooks, reference books and circulate among the members for inputs. Subsequently, in another meeting again held at University of Hyderabad, the detailed syllabus was discussed and finalized incorporating the suggestions of members. The Qualification Descriptors (QD), Program Learning Outcomes (PLO) and the Course Learning Outcomes (CLO) were also finalized keeping the broad requirement of the programme in view. The LOCF also gives general guidelines for the Teaching Learning Process (TLP) corresponding to each component of theory, experiment, tutorials, projects and industrial / field visits to be followed in order to achieve the stated outcomes for each component. Finally, some suggestions for using various methods in the assessment and evaluation of learning levels of students are also made.

The main objective of this whole exercise is to prepare a comprehensive course structure with detailed syllabus along with quality reading material in order to have a uniform standard of education in undergraduate Computer Science programme in the country. This document shall serve as a model document across the higher education institutes (HEIs) in the country for teachers, students and academic administrators. It is a student centric framework where they are expected to learn fundamentals of computer science along with the latest trends and techniques like Artificial Intelligence, Internet of Things, Machine Intelligence alongwith advanced skillsets that include Mobile Application Development, Object Oriented Programming among many other courses.

We sincerely hope that our sincere effort in this endeavor will help the students to be equipped with fundamental as well as advanced and latest technologies in computer science after completion of the programme irrespective of the location and institute across the length and breadth of the country. This will also prepare to opt for higher education in top notch universities and institutes within and outside India. We thank UGC and other experts who contributed in our endeavor.

1. Introduction

Computer Science (CS) has been evolving as an important branch of science and engineering throughout the world in last couple of decades and it has carved out a space for itself like any other disciplines of basic science and engineering. Computer science is a discipline that spans theory and practice and it requires thinking both in abstract terms and in concrete terms. Nowadays, practically everyone is a computer user, and many people are even computer programmers. Computer Science can be seen on a higher level, as a science of problem solving and problem solving requires precision, creativity, and careful reasoning. The ever-evolving discipline of computer science also has strong connections to other disciplines. Many problems in science, engineering, health care, business, and other areas can be solved effectively with computers, but finding a solution requires both computer science expertise and knowledge of the particular application domain.

Computer science has a wide range of specialties. These include Computer Architecture, Software Systems, Graphics, Artificial Intelligence, Computational Science, and Software Engineering. Drawing from a common core of computer science knowledge, each specialty area focuses on specific challenges. Computer Science is practised by mathematicians, scientists and engineers. Mathematics, the origins of Computer Science, provides reason and logic. Science provides the methodology for learning and refinement. Engineering provides the techniques for building hardware and software.

Universities and other HEIs introduced programmes of studies in computer science as this discipline evolved itself to a multidisciplinary discipline. Information Technology is growing rapidly. Increasing applications of computers in almost all areas of human endeavour has led to vibrant industries with concurrent rapid change in technology. Unlike other basic disciplines, developing core competency in this discipline that can be reasonably stable becomes a challenge. In India, it was initially introduced at the Master (postgraduate) level as MCA and M.Tech. Later on, engineering programmes such as B.Tech and B.E in Computer Science & Engineering and in Information Technology were introduced in various engineerin g College/Institutions to cater to the growing demand for trained engineering manpower in IT industries. Parallelly, BSc and MSc programmes with specialisation in Computer Science were introduced to train manpower in this

highly demanding area. B.Sc and B.Sc(Hons) in Computer Science are being planned and introduced in different colleges and institutions.

Computer Science education at undergraduate level (+3) will result in earning a Bachelor of Arts (BA) or Bachelor of Science (BS) degree in CS. The coursework required to earn a BSc is equally weighted in mathematics and science. B.Sc with CS and BSc(Hons) in CS are aimed at undergraduate level training facilitating multiple career paths. Students so graduated, can take up postgraduate programmes in CS leading to research as well as R&D, can be employable at IT industries, or can pursue a teachers' training programme such BEd in Computer Education, or can adopt a business management career. BSc with CS aims at laying a strong foundation of CS at an early stage of the career along with two other subjects such as Physics, Maths, Electronics, Statistics etc. There are several employment opportunities and after successful completion of an undergraduate programme in CS, graduating students can fetch employment directly in companies as Web Developer, Software Engineer, Network Administrator, Data Scientist, or AI/ML personnel.

The Learning Outcome-based Curriculum Framework in Computer Science is aimed at allowing flexibility and innovation in design and development of course content, in method of imparting training, in teaching learning process and in assessment procedures of the learning outcomes. The emphasis in computer science courses, in outcome-based curriculum framework, help students learn solving problems, accomplishing IT tasks, and expressing creativity, both individually and collaboratively. The proposed framework will help Students learn programming techniques and the syntax of one or more programming languages.

Many of the learning outcomes of Computer Science can be achieved only by programming a computer for several different meaningful purposes. All students must, therefore, have access to a computer with a modern programming language installed. The computer science framework does not prescribe a specific language. The teacher and students will decide which modern programming languages students will learn. More importantly, students will learn to adapt to changes in programming languages and learn new languages as they are developed.

The present Learning Outcome-based Curriculum Framework for bachelor's degrees in CS is intended to facilitate the students to achieve the following.

- To develop an understanding and knowledge of the basic theory of Computer Science and Information Technology with good foundation on theory, systems and applications such as algorithms, data structures, data handling, data communication and computation.
- To develop the ability to use this knowledge to analyse new situations
- To acquire necessary and state-of-the-art skills to take up industry challenges. The objectives and outcomes are carefully designed to suit to the above-mentioned purpose.
- The ability to synthesize the acquired knowledge, understanding and experience for a better and improved comprehension of the real-life problems
- To learn skills and tools like mathematics, statistics, physics and electronics to find the solution, interpret the results and make predictions for the future developments.

2. Curriculum Planning- Learning Outcomes-based Approach

2.1 Nature and Extent of the B.Sc/B.Sc. (Hons.) Programme

The undergraduate programs in Computer Science builds on science-based education at +2 level. The +2 senior secondary school education aims and achieves a sound grounding in understanding the basic scientific temper with introduction to process of computation by introducing some programming languages. This prepares a young mind to launch a rigorous investigation of exciting world of computer science.

Framing and implementation of curricula and syllabi is envisaged to provide an understanding of the basic connection between theory and experiment and its importance in understanding the foundation of computing. This is very critical in developing a scientific temperament and to venture a career which a wide spectrum of applications as well as theoretical investigations. The undergraduate curriculum provides students with theoretical foundations and practical experience in both hardware and software aspects of computers. The curriculum in computer science is integrated with courses in the sciences and the humanities to offer an education that is broad, yet of enough depth and relevance to enhance student employment opportunities upon graduation. As a Bachelor's degree program, the curriculum is based on the criterion that graduates are expected to function successfully in a professional employment environment immediately upon graduation.

The undergraduate program in Computer Science is presently being offered though the courses designed for granting the following degrees by various colleges and universities in India. All the courses are of 3-year duration spread over six semesters.

i. B.Sc (Honours) Computer Science
ii. B.Sc with Computer Science

B. Sc. with Computer Science

B.Sc. or Bachelor of Science with Computer Science is a general multidiscipline bachelor programme. The programme has a balanced emphasis on three science subjects, one of which is computer science. A student studying B.Sc. with Computer Science is required to choose two other subjects from a pool of subjects which include Physics,

Mathematics, Statistics, Electronics, Chemistry. Different institutions offer different choice of combinations of subjects. Most popular combinations are Physics and Mathematics, Physics and Electronics, Mathematics and Electronics, but there are also combinations like Statistics and Economics or Commerce and Economics alongwith Computer Science.

B.Sc.(Hons) in Computer Science

B.Sc. (Hons) in India is generally a three-year degree program which develops advanced theoretical and research skills in subject in which Honours is opted. It is a specialized programme offering specialization in one science subject and another auxiliary science subject. This programme helps in building an advanced professional or academic career. It is an appropriate course for students who wish to pursue a Master of Science (M.Sc) or Doctor of Philosophy (PhD) and a research or academic career. This program facilitates students who wish to pursue an independent research project in an area of interest under the supervision of an academic. B.Sc.(Hons) differs from BSc in the number of courses in the subject in which Honours is opted. Thus BSc(Hons) has more CS courses than that of BSc programme.

B.Sc. with CS and B.Sc. (Hons) in CS follow CBCS structure as mandated by UGC. In accordance with CBCS guidelines the courses are categorized into compulsory courses, elective courses, ability enhancement courses. These categories of courses are discussed below keeping the present context of undergraduate education in CS in mind.

2.2 Types of Courses

2.2.1 Core Course (CC)

A core course is a mandatory course required in degree. **Core course** of study refers to a series or selection of courses that all students are required to complete before they can move on to the next level in their education or earn a diploma. The general educational purpose of a core course of study is to ensure that all students take and complete courses that are academically and culturally essential. These are the courses that teach students the foundational knowledge and skills they will need in securing the specific degree or diploma. The core courses are designed with an aim to cover the basics that is expected of a student to imbibe in that particular

discipline. Thus, a course, which should compulsorily be studied by a candidate as a core requirement is termed as a Core course. The present document specifies the core courses for B.Sc. The courses (papers, as referred popularly) under this category are going to be taught uniformly across all universities with 30% deviation proposed in the draft. The purpose of fixing core papers is to ensure that all the institutions follow a minimum common curriculum so that each institution/ university adheres to common minimum standard.

2.2.2 Electives

Generally a course which can be chosen from a pool of courses and which may be very specific or specialized or advanced or supportive to the discipline/ subject of study or which provides an extended scope or which enables an exposure to some other discipline/subject/domain or nurtures the candidate's proficiency/skill is called an Elective Course. Different types of elective courses mandated in the present framework are the following.

- Domain Specific Elective (DSE)
- Generic Elective (GE)
- Ability Enhancement Elective (AEEC)

2.2.3 Discipline Specific Elective (DSE)

Elective courses offered under the main discipline/subject of study is referred to as Discipline Specific Elective. The list provided under this category are suggestive in nature and HEI has freedom to suggest its own papers under this category based on their expertise, specialization, requirements, scope and need. The University/Institute may also offer discipline related elective courses of interdisciplinary nature (to be offered by main discipline/subject of study).

2.2.4 Generic Elective (GE)

An elective course chosen from another discipline/subject, with an intention to seek exposure beyond discipline/s of choice is called a Generic Elective. The purpose of this category of papers is to offer the students the option to explore disciplines of interest beyond the choices they make in Core and Discipline Specific Elective papers. The list provided under this category are suggestive in nature and HEI can design its own papers under this category based on available expertise, specialization, and contextual requirements, scope and need.

P.S.: A core course offered in a discipline/subject may be treated as an elective by other discipline/subject and vice versa and such electives may also be referred to as Generic Elective.

GE for B.Sc. Honours- ONE auxiliary discipline of interest other than major subject in which Honours is opted from a set of related science disciplines is chosen for the entire 3-year and Generic Elective (GE) are opted one paper for each semester in the chosen discipline only.

2.2.5 Dissertation/Project

An elective course designed to acquire special/advanced knowledge, such as supplement study/support study to a project work, and a candidate studies such a course on his/her own with an advisory support by a teacher/faculty member is called dissertation/project.

2.2.6 Ability Enhancement Courses (AEC)

The Ability Enhancement Courses may be of two kinds:

A. **Ability Enhancement Compulsory Courses (AECC):** AECC are the courses based upon the content that leads to knowledge enhancement. These are mandatory for all disciplines. Ability Enhancement Compulsory Courses (AECC) are the following.

- AECC-I English
- AECC-II English/Hindi/ MIL Communications
- AECC-III Environment Science

B. Skill Enhancement Courses (SEC): SEC courses are value-based and/or skill-based and are aimed at providing hands-on-training, competencies, skills, etc. SEC are at least 2 courses for Honours courses and 4 courses for General bachelor programmes. These courses may be chosen from a pool of courses designed to provide value-based and/or skill-based knowledge and should contain both theory and lab/hands-on/training/field work. The main purpose of these courses is to provide students life-skills in hands-on mode to increase their employability. The list provided under this category are suggestive in nature and each university has freedom to suggest their own papers under this category based on their expertise, specialization, requirements, scope and need.

2.2.7 Practical/Tutorial

For each core course and DSE course there will be one practical. The list of practical provided is suggestive in nature and each university has the freedom to add/subtract/edit practical from the list depending on their faculty and infrastructure available. Addition will however be of similar nature.

2.3 Aims of Bachelor of Science Programmes in Computer Science

The Bachelor of Science degree in Computer Science emphasizes problem solving in the context of algorithm development and software implementation and prepares students for effectively using modern computer systems in various applications. The curriculu m provides required computer science courses such as programming languages, data structures, computer architecture and organization, algorithms, database systems, operating systems, and software engineering; as well as elective courses in artificial intelligence, computer-based communication networks, distributed computing, information security, graphics, human-computer interaction, multimedia, scientific computing, web technology, and other current topics in computer science. The main aim of this Bachelor's degree is to deliver a modern curriculum that will equip graduates with strong theoretical and practical backgrounds to enable them to excel in the workplace and to be lifelong learners. The purpose of the BS programs in computer science are twofold: (1) to prepare the student for a position involving the design, development and implementation of computer software/hardware, and (2) to prepare the student for entry into a program of postgraduate study in computer science/engineering and related fields.

The Bachelor of Science program with Computer Science as one subject (BSc with CS) and the Bachelor of Science Honours programme in Computer Science (BSc(Hons) in CS) focus on the concepts and techniques used in the design and development of software systems. Students in this program explore the conceptual underpinnings of Computer Science -- its fundamental algorithms, programming languages, operating systems, and software engineering techniques. In addition, students choose from a rich set of electives that includes data science, computer graphics, artificial intelligence, database systems, computer architecture, and computer networks, among other topics. A generous allotment of free electives allows students to combine study in computer science with study in auxiliary fields to formulate a program that combines experiences across disciplines.

3. Graduate Attributes

Graduate Attributes (GA) are the qualities, skills and understandings that students should develop during their time with the HEI. These are qualities that also prepare graduates as agents of social good in future. Graduate Attributes can be viewed as qualities in following subcategories.

- Knowledge of the discipline
- Creativity
- Intellectual Rigour
- Problem Solving and Design
- Ethical Practices
- Lifelong Learning
- Communication and Social Skills

Among these attributes, categories attributes under *Knowledge of the Discipline* are specific to a programme of study.

3.1.a. Knowledge of Discipline of CS

Knowledge of a discipline is defined as "command of a discipline to enable a smooth transition and contribution to professional and community settings. This Graduate Attribute describes the capability of demonstrating comprehensive and considered knowledge of a discipline. It enables students to evaluate and utilise information and apply their disciplinary knowledge and their professional skills in the workplace.

3.1.b. Creativity

Creativity is a skill that underpins most activities, although this may be less obvious in some disciplines. Students are required to apply imaginative and reflective thinking to their studies. Students are encouraged to look at the design or issue through differing and novel perspectives. Creativity allows the possibility of a powerful shift in outlook and enables students to be open to thinking about different concepts and ideas.

3.1.c. Intellectual Rigour

Intellectual Rigour is the commitment to excellence in all scholarly and intellectual activities, including critical judgement. The students are expected in having clarity in thinking. This capability involves engaging constructively and methodically when exploring ideas, theories and philosophies. It also relates to the ability to analyse and construct knowledge with depth, insight and intellectual maturity.

3.1.d. Problem Solving and Design

Problem solving skills empower students not only within the context of their programmes, but also in their personal and professional lives. Many employers cite good problem

solving skills as a desired attribute that they would like graduates to bring to the workplace. With an ability to seek out and identify problems, effective problem solvers are able to actively engage with a situation, think creatively, to consider different perspectives to address identified challenge, to try out possible solutions and subsequently evaluate results as a way to make decisions. Through this process they can consolidate new and emergent knowledge and develop a deeper understanding of their subject discipline.

3.1.e. Ethical Practices

Ethical practice is a key component of professionalism and needs to be instilled in curricula across courses. When operating ethically, graduates are aware that we live in a diverse society with many competing points of view. Ethical behaviour involves tolerance and responsibility. It includes being open-minded about cultural diversity, linguistic difference, and the complex nature of our world. It also means behaving appropriately towards colleagues and the community and being sensitive to local and global social justice issues.

3.1.f. Life-Long Learning

The skill of being a lifelong learner means a graduate is open, curious, willing to investigate, and consider new knowledge and ways of thinking. This flexibility of mind means they are always amenable to new ideas and actively seek out new ways of learning or understanding the world.

3.1.g. Communication and Social Skills

The ability to communicate clearly and to work well in a team setting is critical to sustained and successful employment. Good communication and social skills involve the ability to listen to, as well as clearly express, information back to others in a variety of ways - oral, written, and visual - using a range of technologies.

3.1.h. Self-Management

Graduates must have capabilities for self-organisation, self-review, personal development and life-long learning.

3.2 LIST OF GRADUATE ATTRIBUTES for B.Sc. and B.Sc.(Hons)

Afore-mentioned GAs can be summarized in the following manner.

GA 1. A commitment to excellence in all scholarly and intellectual activities, including critical judgement

GA 2. Ability to think carefully, deeply and with rigour when faced with new knowledge and arguments.

GA 3. Ability to engage constructively and methodically when exploring ideas, theories and philosophies

GA 4. Ability to consider other points of view and make a thoughtful argument

GA 5. Ability to develop creative and effective responses to intellectual, professional and social challenges

GA 6. Ability to apply imaginative and reflective thinking to their studies

GA 7. Commitment to sustainability and high ethical standards in social and professional practices.

GA 8. To be open-minded about cultural diversity, linguistic difference, and the complex nature of our world

GA 9. Ability to be responsive to change, to be inquiring and reflective in practice, through information literacy and autonomous, self-managed learning.

GA 10. Ability to communicate and collaborate with individuals, and within teams, in professional and community settings

GA 11. Ability to communicates effectively, comprehending and writing effective reports and design documentation, summarizing information, making effective oral presentations and giving and receiving clear oral instructions

GA 12. Ability to demonstrates competence in the practical art of computing in by showing in design an understanding of the practical methods, and using modern design tools competently for complex real-life IT problems

GA 13. Ability to use a range of programming languages and tools to develop computer programs and systems that are effective solutions to problems.

GA 14. Ability to understand, design, and analyse precise specifications of algorithms, procedures, and interaction behaviour.

GA 15. Ability to apply mathematics, logic, and statistics to the design, development, and analysis of software systems

GA 16. Ability to be equipped with a range of fundamental principles of Computer Science that will provide the basis for future learning and enable them to adapt to the constant rapid development of the field.

GA 17. Ability of working in teams to build software systems.

GA 18. Ability to identify and to apply relevant problem-solving methodologies

GA 19. Ability to design components, systems and/or processes to meet required specifications

GA 20. Ability to synthesise alternative/innovative solutions, concepts and procedures

GA 21. Ability to apply decision making methodologies to evaluate solutions for efficiency, effectiveness and sustainability

GA 22. A capacity for self-reflection and a willingness to engage in self-appraisal

GA 23. Open to objective and constructive feedback from supervisors and peers

GA 24. Able to negotiate difficult social situations, defuse conflict and engage positively in purposeful debate.

4. Qualification Descriptors

Qualification descriptors are generic statements of the outcomes of study. Qualification descriptors are in two parts. The first part is a statement of outcomes, achievement of which a student should be able to demonstrate for the award of the qualification. This part will be of interest to those designing, approving and reviewing academic programmes. They will need to be satisfied that, for any programme, the curriculum and assessments provide all students with the opportunity to achieve, and to demonstrat e achievement of, the outcomes. The second part is a statement of the wider abilities that the typical student could be expected to have developed. It will be of assistance to employers and others with an interest in the general capabilities of holders of the qualification. The framework has the flexibility to accommodate diversity and innovation, and to accommodate new qualifications as the need for them arises. It should be regarded as a framework, not as a straitjacket.

4.1. Qualification Descriptor for B.Sc. with CS

On completion of B.Sc. with Computer Science, the expected learning outcomes that a student should be able to demonstrate are the following.

QD-1. Fundamental understanding of the principles of Computer Science and its connections with other disciplines

QD-2. Procedural knowledge that creates different types of professionals related to Computer Science, including research and development, teaching and industry, government and public service;

QD-3. Skills and tools in areas related to computer science and current developments in the academic field of study.

QD-4. Use knowledge, understanding and skills required for identifying problems and issues, collection of relevant quantitative and/or qualitative data drawing on a wide range of sources, and their application, analysis and evaluation using methodologies as appropriate to Computer Science for formulating solutions

QD-5. Communicate the results of studies undertaken in Computer Science accurately in a range of different contexts using the main concepts, constructs and techniques

QD-6. Meet one's own learning needs, drawing on a range of current research and development work and professional materials

QD-7. Apply Computer Science knowledge and transferable skills to new/unfamiliar contexts,

QD-8. Demonstrate subject-related and transferable skills that are relevant to industry and employment opportunities.

4.2. Qualification Descriptors for BSc(Hons) in Computer Science

On completion of B.Sc (Hons) in Computer Science, the expected learning outcomes that a student should be able to demonstrate

QD-Hons 1. A systematic, extensive and coherent knowledge and understanding of the field of computer science as a whole and its applications, and links to related disciplinary areas; including a critical understanding of the established theories, principles and concepts, and of a number of advanced and emerging issues in the field of Computer Science

QD-Hons 2. Procedural knowledge that creates different types of professionals related to Computer Science, including research and development, teaching industry and government and public service;

QD-Hons 3. Skills in areas related to computer science and usage of tools and current developments, including a critical understanding of the latest developments in the area, and an ability to use established techniques of analysis and enquiry within the area of Computer Science.

QD-Hons 4. Demonstrate comprehensive knowledge, including current research, scholarly, and/or professional literature, relating to essential and advanced learning areas pertaining to the chosen disciplinary areas (s) and field of study, and techniques and skills required for identifying problems and issues relating to the disciplinary area and field of study.

QD-Hons 5. Demonstrate skills in identifying information needs, collection of relevant quantitative and/or qualitative data drawing on a wide range of sources, effective analysis and interpretation of data

QD-Hons 6. Use knowledge, understanding and skills for critical assessment of a wide range of ideas and complex problems and issues relating to the chosen field of study.

QD-Hons 7. Communicate the results of studies accurately in a range of different contexts using the main concepts, constructs and techniques of the subject(s) of study;

QD-Hons 8. Address one's own learning needs relating to current and emerging areas of study, making use of research, development and professional materials as appropriate

QD-Hons 9. Apply one's disciplinary knowledge and transferable skills to new/unfamiliar contexts and to identify and analyse problems and issues and seek solutions to real-life problems.

QD-Hons 10. Demonstrate subject-related and transferable skills that are relevant to industry and employment opportunities.

5. Programme Learning Outcomes

These outcomes describe what students are expected to know and be able to do by the time of graduation. They relate to the skills, knowledge, and behaviours that students acquire in their graduation through the program

5.1. Programme Learning Outcomes for BSc with CS

The Bachelor of Science with Computer Science (BSc with CS) program enables students to attain, by the time of graduation:

PLO-A. Demonstrate the aptitude of Computer Programming and Computer based problem solving skills.

PLO-B. Display the knowledge of appropriate theory, practices and tools for the specification, design, implementation

PLO-C. Ability to learn and acquire knowledge through online courses available at different MOOC Providers.

PLO-D. Ability to link knowledge of Computer Science with other two chosen auxiliary disciplines of study.

PLO-E. Display ethical code of conduct in usage of Internet and Cyber systems.

PLO-F. Ability to pursue higher studies of specialization and to take up technical employment.

PLO-G. Ability to formulate, to model, to design solutions, procedure and to use software tools to solve real world problems and evaluate .

PLO-H. Ability to operate, manage, deploy, configure computer network, hardware, software operation of an organization.

PLO-I. Ability to present result using different presentation tools.

PLO-J. Ability to appreciate emerging technologies and tools.

5.2. Additional PLOs for B.Sc. (Hons) in CS

The Bachelor of Science Honours in Computer Science (B.Sc. (Hons) in CS) program enables students to attain following additional attributes besides the afore-mentioned attributes, by the time of graduation:

PLO-K. Apply standard Software Engineering practices and strategies in real-time software project development

PLO-L. Design and develop computer programs/computer -based systems in the areas related to algorithms, networking, web design, cloud computing, IoT and data analytics.

PLO-M. Acquaint with the contemporary trends in industrial/research settings and thereby innovate novel solutions to existing problems

PLO-N. The ability to apply the knowledge and understanding noted above to the analysis of a given information handling problem.

PLO-O. The ability to work independently on a substantial software project and as an effective team member.

6. Course Structures

6.1 Structure of B.Sc. with CS

The B.Sc. programme with CS as one of the subjects consists of 132 credits in accordance with the Choice Based Credit System (CBCS) approved by the UGC with 1 weekly-contact-hour for each credit for theory/tutorials and 2 weekly-contact-hours for each credit of laboratory work.

6.1.1. Credit-wise Distribution - Out of 132 credits, 108 credits are equally divided among CS (denoted as A in the following table) and two other auxiliary subjects, denoted as B and C, (36 credits each). 36 credits for each subject are further distributed as 24 credits for Core Compulsory Courses (CC) and 12 credits for Discipline Specific Electives (DSE). There are 8 credits for Ability Enhancement Compulsory Courses. SEC's will have 16 credits.

6.1.2. Course-wise Distribution - There are 4 CC courses for each subject (CS and two auxiliary subjects). Each CC course is of 6 credits (4 Theory + 2 Practicum). Similarly, there are 2 DSE papers, each of 6 credits. There are 4 Skill Enhancement Courses (SEC) each of 4 credits with a total of 16 credits. 16 credits of SEC are distributed as 8 credits (2 courses) for subject A (CS) and 4 credits for each of two auxiliary subjects, subjects B and C (one courses for each subject). There are two AECC namely, Environ mental Sciences and Languages/ Communications with 4 credits.

6.1.3. Semester-wise Distribution – BSc with CS is a 3-Yr programme with 6 semesters. In each semester, there will be 22 credits. For each of first four semesters, there will be 3 CC, one each for subjects A, B and C accounting to 18 credits. Similarly, for semesters 5 and 6, there will be 3 DSE in each semester and one DSE for each of three subjects (a, B and C). Two AECC will be offered in first two semesters. SEC will be offered in semesters 3, 4, 5 and 6 and a student is required to take any one SEC from a pool of options. However, in semesters 3 and 4, SEC for the auxiliary subjects will be offered and in semesters 5 and 6, SEC for CS will be offered.

The scope of the present proposal is to design CS courses. There are 4 CC courses for CS, 2 DSE courses and 2 SEC (CS related elective). A student can take more than 132 credits in total (but not more than 148 credits) to qualify for the grant of the B.Sc. (CS) degree after completing them successfully as per rules and regulations of the HEI.

Table I presents the structure in a schematic form. Table II gives details of CS papers in each of different course-categories.

6.2 Course Structure of B.Sc (Hons) in Computer Science

The B.Sc. (Hons) in Computer Science programme consists of 148 credits in accordance with UGC's CBCS with 1 contact-hour per week for each credit for Theory/Tutorial and 2 contact-hour per week for each credit for practical.

6.2.1. Credit-wise Distribution – Out of 148 credits in BSc(Hons) in CS, 84 credits of CC papers in CS, 24 credits of GE papers are devoted to one auxiliary subject, DSE paperss are of 24 credits, AECC papers are 8 credits, and SEC papers are of 8 credits.

6.2.2. Course-wise Distribution - In BSc(Hons) in CS, there are 14 core compulsory courses (CC) in CS subjects, each of 6 credits (4+2). There are 4 Discipline Specific Electives (DSE) papers each of 6 (4+2) credits. In addition, there are 2 AEC papers and 2 SEC papers. There are 4 GE papers each of 6 credits. One auxiliary discipline of interest from Mathematics Statistics, Operational Research , Physics, and Electronics for the entire 3-year and Generic Elective (GE) are opted one paper for each semester in the chosen discipline only.

6.2.3. Semester-wise Distribution- Unlike BSC programme, BSc (Hons) programme has uneven distribution of credits in 6 semesters. First two semesters have 22 credits each, third and fourth semesters have 28 credits each and each of fifth and sixth semester has 24 credits.

Table III presents the structure of BSc(Hons) in CS and Table IV lists the CS-specific courses for the programme

.

TABLE I: COURSE STRUCTURE FOR GENERAL B.Sc.

SEMESTER	Compulsory Core Courses (CC) each with 06 credit; 04 Core courses are compulsory for each subject A, B and C	Discipline Specific Elective (DSE) A- for CS; B and C are other subjects	Ability Enhancement Compulsory Courses (AECC)	Skill Enhancement Course (SEC) Select any one in each semester of Sem-III and Sem-IV	Total Credit
Sem- I	CC-1A **CS** CC- 1B *Auxiliary Sub* CC- 1C *Auxiliary Sub*		AECC-1		22
Sem-II	CC-2A **CS** CC-2B *Auxiliary Sub* CC- 2C *Auxiliary Sub*		AECC-2		22
Sem-III	CC-3A **CS** CC- 3B: *Auxiliary Sub* CC- 3C: *Auxiliary Sub*			Any one of the following SEC-1B *Auxiliary* SEC-1C *Auxiliary*	22
Sem-IV	CC-4A **CS** CC- 4B *Auxiliary Sub* CC- 4C *Auxiliary Sub*			Any one of the following SEC-2B *Auxiliary* SEC-2C *Auxiliary*	22
Sem-V		DSE-1A **CS** DSE-1B *Auxiliary* DSE-1C *Auxiliary*		Any one elective of SEC-3A **CS**	22
Sem- VI		DSE-2A **CS** DSE-2B *Auxiliary* DSE-2C *Auxiliary*		Any one elective of SEC-4A **CS**	22

TABLE II: CS COURSE DETAILS FOR GENERAL B.Sc. WITH CS

Course-Type	Course-code as referred above	Compulsory/Elective	List of compulsory courses and list of option of elective courses. (A suggestive list)
CC	CC-1A, CC-2A, CC-3A, CC-4A	Compulsory	Programming Methodologies, AND Data Structure, AND Operating Systems, AND DBMS
DSE	DSE 1A	Elective	Software Engineering, OR Computer Ethics OR Computer Organization & Architecture OR Computer Networks
	DSE 2A	Elective	Data Mining OR Internet of Things OR Artificial Intelligence OR Computer Graphics
SEC	SEC 3A	Elective	MATLAB Programming OR, Programming in Java OR Python Programming
	SEC 4A	Elective	Web Programming OR Mobile Application Development OR Cloud Computing
AEC	AECC1, AECC2	Compulsory	Communication in English, Environmental Science

TABLE III: COURSE STRUCTURE FOR B.Sc. (Hons)

SEM	Core Courses (CC) each with 06 credit. All 14 courses are compulsory	Ability Enhancement Compulsory Courses (AECC) Select any 2 (04 credits each)	Skill Enhancement Course (SEC) Select any 4 courses (04 credits each)	Discipline Specific Elective (DSE) Select any 4 courses (06 credits each)	Generic Elective, 4 courses (06 credits each) in 4 semesters on one auxiliary subject	Total Credit
I	CC-1	AEC-1			GEC-1	22
	CC-2					
II	CC-3	AEC-2			GEC-2	22
	CC-4					
III	CC-5		SEC-1		GEC-3	28
	CC-6					
	CC-7					
IV	CC-8		SEC-2		GEC-4	28
	CC-9					
	CC-10					
V	CC-11			DSE-1		24
	CC-12			DSE-2		
VI	CC-13			DSE-3		24
	CC-14			DSE-4		

TABLE IV: CS COURSE DETAILS FOR B.Sc.(HONS) in CS

SEM	Core Courses (CC) each with 06 credit. 14 compulsory courses	Discipline Specific Electives. 4 courses. One from set of courses in a box	Skill Enhancement Courses SEC 2 courses
I	Programming Methodology		
	Computer System Architecture		
II	Data Structure		
	Discrete Structures		
III	Operating System		Any **one** MATLAB Programming, Programming in Java, Python Programming
	Algorithms		
	Computer Networks		
IV	Software Engineering		Any **one** Mobile Application Dev, Web Programming, GIMP(GNU Image Manipulation Program)
	DBMS		
	Object Oriented Programming		
V	Internet Technologies	Any **two** of (suggestive list) Image Processing, Data Analytics, Computer Ethics, System Security, Human Computer Interface,	
	Artificial Intelligence		
VI	Computer Graphics	Any **two** of (suggestive list) Modelling and Simulation, Theory of Computation, Data Mining, Cloud Computing, Internet of Things , Institutions can add courses	
	Machine Learning		

CS as Generic Elective

For B.Sc. (Hons) programme with Honours in subjects such as Physics, Mathematics, Statistics, Electronics etc, CS can be one of the auxiliary subjects. The following table gives the details CS courses as Generic Elective for BSc(Hons) in other subjects.

TABLE V: CS COURSE DETAILS AS GEC FOR B.Sc.(HONS) in OTHER SUBJECT

SEM	Generic Elective Courses (GEC) each with 06 credit. 4 Courses	
I	GEC-I	Programming Methodology
II	GEC-II	Data Structure OR Discrete Structures
III	GEC-III	Operating System OR Algorithms OR Computer Networks
IV	GEC-IV	Software Engineering OR DBMS OR Object Oriented Programming

6.3 Course Learning Outcomes, Contents, References

PROGRAMMING METHODOLOGY

1. Learn to develop simple algorithms and flow charts to solve a problem.
2. Develop problem solving skills coupled with top down design principles.
3. Learn about the strategies of writing efficient and well-structured computer algorithms/programs.
4. Develop the skills for formulating iterative solutions to a problem.
5. Learn array processing algorithms coupled with iterative methods.
6. Learn text and string processing efficient algorithms.
7. Learn searching techniques and use of pointers.
8. Understand recursive techniques in programming.

SYLLABUS

A. Theory **4 credits**

UNIT I. Introduction to Programming, Program Concept, Characteristics of Programming, Stages in Program Development, Algorithms, Notations, Design, Flowcharts, Types of Programming Methodologies, Introduction to C++ Programming - Basic Program Structure In C++, Variables and Assignments, Input and Output, Selection and Repetition Statements.

UNIT II. Top-Down Design, Predefined Functions, Programmer -defined Function, Local Variable, Function Overloading, Functions with Default Arguments, Call -By-Value and Call-By-Reference Parameters, Recursion.

UNIT III. Introduction to Arrays, Declaration and Referring Arrays, Arrays in Memory, Initializing Arrays. Arrays in Functions, Multi-Dimensional Arrays.

UNIT IV. Structures - Member Accessing, Pointers to Structures, Structures and Functions, Arrays of Structures, Unions.

UNIT V. Declaration and Initialization, Reading and Writing Strings, Arrays of Strings, String and Function, Strings and Structure, Standard String Library Functions.

UNIT VI. Searching Algorithms - Linear Search, Binary Search. Use of files for data input and output. merging and copy files.

TEXT AND REFERENCE BOOKS

- Problem Solving and Program Design in C, J. R. Hanly and E. B. Koffman, Pearson, 2015.
- Programming and problem solving with C++: brief edition, N. Dale and C. Weems, Jones & Bartlett Learning, 2010.

B. Practicum **2 Credits**

Given the problem statement, students are required to formulate problem, develop flowchart/algorithm, write code, execute and test it. Students should be given assignments on following :

a. To learn elementary techniques involving arithmetic operators and mathematical expressions, appropriate use of selection (if, switch, conditional operators) and control structures

b. Learn how to use functions and parameter passing in functions, writing recursive programs.

2. Write Programs to learn the use of strings and string handling operations.
   a. Problems which can effectively demonstrate use of Arrays. Structures and Union.
   b. Write programs using pointers.
   c. Write programs to use files for data input and output.
   d. Write programs to implement search algorithms.

COMPUTER SYSTEM ARCHITECTURE

1. To make students understand the basic structure, operation and characteristics of digital computer.
2. To familiarize the students with arithmetic and logic unit as well as the concept of the concept of pipelining.
3. To familiarize the students with hierarchical memory system including cache memories and virtual memory.
4. To make students know the different ways of communicating with I/O devices and standard I/Ointerfaces.

SYLLABUS **6 credits**

UNIT I Fundamentals of Digital Electronics: Data Types, Complements, Fixed-Point Representation, Floating-Point Representation, Other Binary Codes, Error Detection Codes, Logic Gates, Boolean Algebra, Map Simplification, Combinational Circuits, Flip-Flops, Sequential Circuits, Registers, Counters, Multiplexer, Demultiplexer, Decoder, Encoder.

UNIT II Register Transfer and Micro operations: Register Transfer Language, Register Transfer, Bus & Memory Transfer, Arithmetic Microoperations, Logic Microoperations, Shift Microoperation.

UNIT III Basic Computer Organization: Instruction codes, Computer Registers, Computer Instructions, Timing & Control, Instruction Cycles, Memory Reference Instruction, Input-Output & Interrupts, Complete Computer Description & Design of Basic Computer.

UNIT IV Processor and Control Unit: Hardwired vs. Micro programmed Control Unit, General Register Organization, Stack Organization, Instruction Format, Data Transfer & Manipulation, Program Control, RISC, CISC, Pipelining – Pipelined datapath and control – Handling Data hazards & Control hazards.

UNIT V Memory and I/O Systems: Peripheral Devices, I/O Interface, Data Transfer Schemes, Program Control, Interrupt, DMA Transfer, I/O Processor. Memory Hierarchy, Processor vs. Memory Speed, High-Speed Memories, Cache Memory, Associative Memory, Interleave, Virtual Memory, Memory Management.

UNIT VI Parallelism: Instruction-level-parallelism – Parallel processing challenges – Flynn's classification – Hardware multithreading – Multicore processors

TEXT BOOKS

- Computer System Architecture, M. Morris Mano, 3rd Edition, Prentice Hall.
- Computer Organization and Design, David A. Patterson and John L. Hennessey, Fifth edition, Morgan Kauffman / Elsevier, 2014.

REFERENCE BOOKS

- Computer Architecture: A Quantitative Approach, John L. Hennessy, David A. Patterson, 4th Edition.
- Computer Organization and Architecture, William Stallings, Prentice Hall.

DATA STRUCTURES

1. To be familiar with fundamental data structures and with the manner in which these data structures can best be implemented; become accustomed to the description of algorithms in both functional and procedural styles
2. To have a knowledge of complexity of basic operations like insert, delete, search on these data structures.
3. Ability to choose a data structure to suitably model any data used in computer applications.
4. Design programs using various data structures including hash tables, Binary and general search trees, heaps, graphs etc.
5. Ability to assess efficiency tradeoffs among different data structure implementations.
6. Implement and know the applications of algorithms for sorting, pattern matching etc.

SYLLABUS

A. Theory **4 credits**

UNIT I. Basic concepts- Algorithm Specification-Introduction, Recursive algorithms, Data Abstraction Performance analysis, Linear and Non Linear data structures, Singly Linked Lists-Operations, Concatenating, circularly linked lists-Operations for Circularly linked lists, Doubly Linked Lists- Operations. Representation of single, two dimensional arrays, sparse matrices-array and linked representations.

UNIT II. Stack- Operations, Array and Linked Implementations, Applications- Infix to Postfix Conversion, Postfix Expression Evaluation, Recursion Implementation, Queue- Definition and Operations, Array and Linked Implementations, Circular Queues - Insertion and Deletion Operations, Dequeue (Double Ended Queue).

UNIT III. Trees, Representation of Trees, Binary tree, Properties of Binary Trees, Binary Tree Representations- Array and Linked Representations, Binary Tree Traversals, Threaded Binary Trees, Priority Queue- Implementation, Heap- Definition, Insertion, Deletion.

UNIT IV. Graphs, Graph ADT, Graph Representations, Graph Traversals, Searching, Static Hashing- Introduction, Hash tables, Hash functions, Overflow Handling.

UNIT V. Sorting Methods, Comparison of Sorting Methods, Search Trees- Binary Search Trees, AVL Trees- Definition and Examples.

UNIT VI. Red-Black and Splay Trees, Comparison of Search Trees, Pattern Matching Algorithm- The Knuth-Morris-Pratt Algorithm, Tries (examples).

TEXTBOOKS

- Fundamentals of Data structures in C, 2nd Edition, E. Horowitz, S. Sahni and Susan Anderson-Freed, Universities Press.

- Data structures and Algorithm Analysis in C, 2nd edition, M. A. Weiss, Pearson.
- Lipschutz: Schaum's outline series Data structures Tata McGraw-Hill

B. Practicum **2 credits**

Students are required to write and practically execute programs to solve problem using various data structures. The teacher can suitably device problems which help students experiment using the suitable datastructures and operations. Some of the problems are indicated below.

1. Write program that uses functions to perform the following:

    a) Creation of list of elements where the size of the list, elements to be inserted and deleted are dynamically given as input.

    b) Implement the operations, insertion, deletion at a given position in the list and search for an element in the list

    c) To display the elements in forward / reverse order

2. Write a program that demonstrates the application of stack operations (Eg: infix expression to postfix conversion)

3. Write a program to implement queue data structure and basic operations on it (Insertion, deletion, find length) and code atleast one application using queues.

4. Write a program that uses well defined functions to Create a binary tree of elements and Traverse the a Binary tree in preorder, inorder and postorder,

5. Write program that implements linear and binary search methods of searching for an elements in a list

6. . Write and trace programs to understand the various phases of sorting elements using the methods

    a) Insertion Sort

    b) Quicksort

    c) Bubble sort

7. Write and trace programs to Create a Binary search tree and insert and delete from the tree.

8. Represent suitably a graph data structure and demonstrate operations of travesrals on it.

DISCRETE STRUCTURES

1. Understand the notion of mathematical thinking, mathematical proofs, and algorithmic thinking, and be able to apply them in problem solving.
2. Understand the basics of combinatorics, and be able to apply the methods from these subjects in problem solving.
3. Be able to use effectively algebraic techniques to analyse basic discrete structures and algorithms.
4. Understand asymptotic notation, its significance, and be able to use it to analyse asymptotic performance for some basic algorithmic examples.
5. Understand some basic properties of graphs and related discrete structures, and be able to relate these to practical examples.

SYLLABUS **6 credits**

UNIT I. Sets: Finite and Infinite Sets, Uncountable Infinite Sets; Functions, Relations, Properties of Binary Relations, Closure, Partial Ordering Relations; Counting - Pigeonhole Principle, Permutation and Combination; Mathematical Induction, Principle of Inclusion and Exclusion.

UNIT II. Growth of Functions: Asymptotic Notations, Summation Formulas and Properties, Bounding Summations, Approximation by Integrals

UNIT III. Recurrences: Recurrence Relations, Generating Functions, Linear Recurrence Relations with Constant Coefficients and their Solution, Substitution Method, Recurrence Trees, Master Theorem

UNIT IV. Graph Theory: Basic Terminology, Models and Types, Multigraphs and Weighted Graphs, Graph Representation, Graph Isomorphism, Connectivity, Euler and Hamiltonian Paths and Circuits, Planar Graphs, Graph Coloring, Trees, Basic Terminology and Properties of Trees, Introduction to Spanning Trees

UNIT V. Propositional Logic: Logical Connectives, Well-formed Formulas, Tautologies, Equivalences, Inference Theory

REFERENCE BOOKS

- C.L. Liu & Mahopatra, Elements of Discrete mathematics, 2nd Sub Edition 1985, Tata McGraw Hill
- Rosen, Discrete Mathematics and Its Applications, Sixth Edition 2006
- T.H. Coremen, C.E. Leiserson, R. L. Rivest, Introduction to algorithms, Prentice Hall on India (3rd edition 2009)
- M. O. Albertson and J. P. Hutchinson, Discrete Mathematics with Algorithms 1988 John wiley Publication

OPERATING SYSTEM

1. Describe the important computer system resources and the role of operating system in their management policies and algorithms.
2. To understand various functions, structures and history of operating systems and should be able to specify objectives of modern operating systems and describe how operating systems have evolved over time.
3. Understanding of design issues associated with operating systems.
4. Understand various process management concepts including scheduling, synchronization, and deadlocks.
5. To have a basic knowledge about multithreading.
6. To understand concepts of memory management including virtual memory.
7. To understand issues related to file system interface and implementation, disk management.
8. To understand and identify potential threats to operating systems and the security features design to guard against them.
9. To have sound knowledge of various types of operating systems including Unix and Android.
10. Describe the functions of a contemporary operating system with respect to convenience, efficiency, and the ability to evolve.

SYLLABUS **6 credits**

UNIT I. (Introduction to Operating System) What is Operating System? History and Evolution of OS, Basic OS functions, Resource Abstraction, Types of Operating Systems– Multiprogramming Systems, Batch Systems, Time Sharing Systems; Operating Systems for Personal Computers, Workstations and Hand-held Devices, Process Control & Real time Systems.

UNIT II. (Operating System Organization and Process Characterization) Processor and User Modes, Kernels, System Calls and System Programs, System View of the Process and Resources, Process Abstraction, Process Hierarchy, Threads, Threading Issues, Thread Libraries; Process Scheduling, Non-Pre-emptive and Pre-emptive Scheduling Algorithms.

UNIT III. Process Management (Deadlock) Deadlock, Deadlock Characterization, Necessary and Sufficient Conditions for Deadlock, Deadlock Handling Approaches: Deadlock Prevention, Deadlock Avoidance and Deadlock Detection and Recovery.

UNIT IV. (Inter Process Communication and Synchronization) Concurrent and Dependent Processes, Critical Section, Semaphores, Methods for Inter-process Communication; Process Synchronization, Classical Process Synchronization Problems: Producer-Consumer, Reader-Writer.

UNIT V. (Memory Management) Physical and Virtual Address Space; Memory Allocation Strategies– Fixed and -Variable Partitions, Paging, Segmentation, Virtual Memory.

UNIT VI. (File and I/O Management, OS security) Directory Structure, File Operations, File Allocation Methods, Device Management, Pipes, Buffer, Shared Memory, Security Policy Mechanism, Protection, Authentication and Internal Access Authorization

UNIT VII. (Introduction to Android Operating System) Introduction to Android Operating System, Android Development Framework, Android Application Architecture, Android Process Management and File System, Small Application Development using Android Development Framework.

REFERENCE BOOKS

- A Silberschatz, P.B. Galvin, G. Gagne, Operating Systems Concepts, 8th Edition, John Wiley Publications 2008.
- A.S. Tanenbaum, Modern Operating Systems, 3rd Edition, Pearson Education 2007.
- G. Nutt, Operating Systems: A Modern Perspective, 2nd Edition Pearson Education 1997.
- W. Stallings, Operating Systems, Internals & Design Principles 2008 5th Edition, Prentice Hall of India.
- M. Milenkovic, Operating Systems- Concepts and design, Tata McGraw Hill 1992.

ALGORITHMS

1. To learn good principles of algorithm design;
2. To learn how to analyse algorithms and estimate their worst-case and average-case behaviour (in easy cases);
3. To become familiar with fundamental data structures and with the manner in which these data structures can best be implemented; become accustomed to the description of algorithms in both functional and procedural styles;
4. To learn how to apply their theoretical knowledge in practice (via the practical component of the course).

SYLLABUS

A Theory **4 Credits**

UNIT I. Introduction: Basic Design and Analysis Techniques of Algorithms, Correctness of Algorithm. Algorithm Design Techniques: Iterative Techniques, Divide and Conquer, Dynamic Programming, Greedy Algorithms.

UNIT II. Sorting and Searching Techniques: Elementary Sorting techniques– Bubble Sort, Insertion Sort, Merge Sort, Advanced Sorting techniques- Heap Sort, Quick Sort, Sorting in Linear Time - Bucket Sort, Radix Sort and Count Sort, Searching Techniques- Medians & Order Statistics, complexity analysis

UNIT III. Graphs Algorithms: Graph Algorithms– Breadth First Search, Depth First Search and its Applications, Minimum Spanning Trees. String Processing

UNIT IV. Lower Bounding Techniques: Decision Trees, Balanced Trees, Red-Black Trees

UNIT V. Advanced Analysis Technique: Randomized Algorithm, Distributed Algorithm, Heuristics

RECOMMENDED BOOKS

- T.H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein Introduction to Algorithms, PHI, 3rd Edition 2009
- Sara basse & A.V. Gelder Computer Algorithm – Introduction to Design and Analysis, Publisher – Pearson 3rd Edition 1999

B. Practicum **2 Credits**

The student shall develop programs in a chosen language to solve problems using algorithm design techniques such asDivide and Conquer, Greedy, Dynamic programming and Backtracking. Some of the problems to be solved are indicated below.

1. Write a test program to implement Divide and Conquer Strategy . Eg: Quick sort algorithm for sorting list of integers in ascending order
2. Write a program to implement Merge sort algorithm for sorting a list of integers in ascending order.
3. Write program to implement the DFS and BFS algorithm for a graph.
4. Write program to implement backtracking algorithm for solving problems like N-queens ..
5. Write a program to implement the backtracking algorithm for the sum of subsets problem
6. Write program to implement greedy algorithm for job sequencing with deadlines.
7. Write a program to implement Dijkstra's algorithm for the Single source shortest path problem.
8. Write a program that implements Prim's algorithm to generate minimum cost spanning tree.
9. Write a program that implements Kruskal's algorithm to generate minimum cost spanning tree
10. Write program to implement Dynamic Programming algorithm for the 0/1 Knapsack problem.
11. Write program to implement Dynamic Programming algorithm for the Optimal Binary Search Tree Problem.

COMPUTER NETWORKS

1. Understand the structure of Data Communications System and its components. Be familiarize with different network terminologies.
2. Familiarize with contemporary issues in network technologies.
3. Know the layered model approach explained in OSI and TCP/IP network models
4. Identify different types of network devices and their functions within a network.
5. Learn basic routing mechanisms, IP addressing scheme and internetworking concepts.
6. Familiarize with IP and TCP Internet protocols.
7. To understand major concepts involved in design of WAN, LAN and wireless networks.
8. Learn basics of network configuration and maintenance.
9. Know the fundamentals of network security issues.

SYLLABUS **6 credits**

UNIT I. Introduction to Computer Networks and Networking Elements: Network Definition, Network Topologies, Network Classifications, Network Protocol, Layered Network Architecture, Overview of OSI Reference Model, Overview of TCP/IP Protocol Suite, Hub, Switch (Managed and Unmanaged), Routers

UNIT II. Data Communication Fundamentals and Techniques: Analog and Digital Signal, Data-Rate Limits, Digital to Digital Line Encoding Schemes, Pulse Code Modulation, Parallel and Serial Transmission, Digital to Analog Modulation - Multiplexing Techniques- FDM, TDM, Transmission Media.

UNIT III. Networks Switching Techniques and Access Mechanisms: Circuit Switching, Packet Switching- Connectionless Datagram Switching, Connection-Oriented Virtual Circuit Switching; Dial-Up Modems, Digital Subscriber Line, Cable TV for Data Transfer.

UNIT IV. Data Link Layer Functions and Protocol: Error Detection and Error Correction Techniques, Data-Link Control- Framing and Flow Control, Error Recovery Protocols-Stop and Wait ARQ, Go-Back-N ARQ, Point to Point Protocol on Internet.

UNIT V. Multiple Access Protocol and Network Layer: CSMA/CD Protocols, Ethernet LANS; Connecting LAN and Back-Bone Networks- Repeaters, Hubs, Switches, Bridges, Router and Gateways, Networks Layer Functions and Protocols (6 Lectures) Routing, Routing Algorithms, Network Layer Protocol of Internet - IP Protocol, Internet Control Protocols.

UNIT VI. Transport Layer and Application Layer Functions and Protocols: Transport Services- Error and Flow Control, Connection Establishment and Release- Three Way Handshake, Overview of Application Layer Protocol (5 Lectures) Overvi ew of DNS Protocol; Overview of WWW & HTTP Protocol.

REFERENCE BOOKS

- B. A. Forouzan: Data Communications and Networking, Fourth edition, THM Publishing Company Ltd 2007.
- A. S. Tanenbaum: Computer Networks, Fourth edition, PHI Pvt. Ltd 2002

SOFTWARE ENGINEERING

1. Basic knowledge and understanding of the analysis and design of complex systems.
2. Ability to apply software engineering principles and techniques.
3. To produce efficient, reliable, robust and cost-effective software solutions.
4. Ability to work as an effective member or leader of software engineering teams.
5. To manage time, processes and resources effectively by prioritising competing demands to achieve personal and team goals Identify and analyzes the common threats in each domain.

SYLLABUS **6 credits**

UNIT I. Software Development Approaches: Introduction; Evolving Role of Software; Software Characteristics; Software Applications. Software Design Processes: Introduction; What is Meant by Software Engineering?, Definitions of Software Engineering; The Serial or Linear Sequential Development Model; Iterative Development Model; The incremental Development Model

UNIT II. Software Design Principles: Introduction, System Models: Data-flow Models, Semantic Data Models, Object Models, Inheritance Models, Object Aggregation, Service Usage Models, Data Dictionaries; Software Design: The Design Process, Design Methods, Design description, Design Strategies, Design Quality; Architectural Design: System Structuring, The Repository Model, The Client–Server Model, The Abstract Machine Model, Control Models, Modular Decomposition, Domain-Specific Architectures.

UNIT III. Object Oriented Design: Introduction; Object Oriented Design: Objects, Object Classes & Inheritance, Inheritance, Object Identification, An Object-Oriented Design Example, Object Aggregation; Service Usage; Object Interface Design: Design Evolution, Function Oriented Design, Data–Flow Design; Structural Decomposition: Detailed Design.

UNIT IV. An Assessment of Process Life-Cycle Models: Introduction; Overview of the Assessment of Process; The Dimension of Time; The Need for a Business Model in Software Engineering; Classic Invalid Assumptions: First Assumption: Internal or External Drivers, Second Assumption: Software or Business Processes, Third Assumption: Processes or Projects, Fourth Assumption: Process Centered or Architecture Centered; Implications of the New Business Model; Role of the Problem-Solving Process in this Approach: Data, Problem Definition, Tools and Capabilities; Redefining the Software Engineering Process: Round-Trip Problem-Solving Approach, Activities, Goals, Interdisciplinary Resources, Time.

UNIT V. Software Reliability: Introduction; Software Reliability Metrics; Programming for Reliability: Fault Avoidance, Fault Tolerance, Software Reuse.

UNIT VI. Software Testing Techniques: Introduction; Software Testing Fundamental; Testing Principles; White Box Testing; Control Structure Testing; Black Box Testing; Boundary Value Analysis; Testing GUIs; Testing Documentation and Help Facilities; Software Testing Strategies: Introduction; Organizing for Software

Testing; Software Testing Strategy, Unit Testing: Unit Test Considerations, Top-Down Integration, Bottom-Up Integration.

REFERENCE BOOKS

- R. G. Pressman – Software Engineering, TMH
- Sommerville, Ian, Software Engineering, Pearson Education
- Pankaj Jalote – An Integrated Approach to Software Engineering, Narosa Publications.
- Pfleeger, Shari Lawrence, Software Engineering Theory and Practice, second edition. Prentice- Hall 2001.
- Object Oriented & Classical Software Engineering (Fifth Edition), SCHACH,TMH

DATABASE MANAGEMENT SYSTEMS

1. Gain knowledge of database systems and database management systems software.
2. Ability to model data in applications using conceptual modelling tools such as ER Diagrams and design data base schemas based on the model.
3. Formulate, using SQL, solutions to a broad range of query and data update problems.
4. Demonstrate an understanding of normalization theory and apply such knowledge to the normalization of a database.
5. Be acquainted with the basics of transaction processing and concurrency control.
6. Familiarity with database storage structures and access techniques.
7. Compare, contrast and analyse the various emerging technologies for database systems such as NoSQL.
8. Analyse strengths and weaknesses of the applications of database technologies to various subject areas.

SYLLABUS

A Theory **4 Credits**

UNIT I. Basic Database Concepts, Terminology, and Architecture; Types of Database Management Systems. Differences between Relational and other Database Models. Data Modelling: Relations, Schemas, Constraints, Queries, and Updates; Conceptual vs. Physical Modeling; Entity Types, attributes, ER Diagrams.

UNIT II. SQL Data Definition: Specifying Tables, Data Types, Constraints; Simple SELECT, INSERT, UPDATE, DELETE Statements; Complex SELECT Queries, including Joins and Nested Queries; Actions and Triggers; Views; Altering Schemas.

UNIT III. Relational Algebra: Definition of Algebra; Relations as Sets; Operations: SELECT, PROJECT, JOIN, etc. Normalization Theory and Functional Dependencies, 2NF, 3NF, BCNF, 4NF, 5NF;

UNIT IV. Indexing: Files, Blocks, and Records, Hashing; RAID; Replication; Single-Level and Multi-Level Indexes; B-Trees and B+-Trees. Query Processing Translation of SQL into Query Plans; Basics of Transactions, Concurrency and Recovery.

UNIT V. DATABASE PROGRAMMING: Embedded SQL; Dynamic SQL, JDBC; Avoiding Injection Attacks; Stored Procedures; Lightweight Data Access Layers for Python and JavaScript Applications; PHP and MySQL, Object Relational Modeling: Hibernate for Java, Active Record for Rails.

UNIT VI. BIG DATA: Motivations; OLAP vs. OLTP; Batch Processing; MapReduce and Hadoop; Spark; Other Systems: HBase. Working with POSTGRES, REDIS, MONGO, and NEO: Setting up the same Database on Four Platforms; Basic Queries and Reporting.

TEXTBOOKS

- Elmasri's and Navathe's *Fundamentals of Database Systems*. Addison-Wesley

REFERENCE BOOK

- Data base Management Systems, Raghu Ramakrishnan, Johannes Gehrke, McGraw Hill Education
- Data base System Concepts, A. Silberschatz, Henry. F. Korth, S. Sudarshan, McGraw Hill Education

B. Practicum **2 credits**

Students are required to practice the concepts learnt in the theory by designing and querying a database for a chosen organization (Like Library, Transport etc). The teacher may devise appropriate weekly lab assignments to help students practice the designing , querying a database in the context of example database. Some indicative list of experiments is given below.

Experiment 1: E-R Model

Analyze the organization and identify the entities , attributes and relationships in it. .

Identify the primary keys for all the entities. Identify the other keys like candidate keys, partial keys, if any.

Experiment 2: Concept design with E-R Model

Relate the entities appropriately. Apply cardinalities for each relationship. Identify strong entities and weak entities (if any).

Experiment 3: Relational Model

Represent all the entities (Strong, Weak) in tabular fashion. Represent relationships in a tabular fashion.

Experiment 4: Normalization

Apply the First, Second and Third Normalization levels on the database designed for the organization

Experiment 5: Installation of Mysql and practicing DDL commands

Installation of MySql. Creating databases, How to create tables, altering the database, dropping tables and databases if not required. Try truncate, rename commands etc.

Experiment 6: Practicing DML commands on the Database created for the example organization

DML commands are used to for managing data within schema objects. Some examples:

- SELECT - retrieve data from the a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - deletes all records from a table, the space for the records remain

Experiment 7: Querying

practice queries (along with sub queries) involving ANY, ALL, IN, Exists, NOT EXISTS, UNION, INTERSECT, Constraints etc.

Experiment 8 and Experiment 9: Querying (continued…)

Practice queries using Aggregate functions (COUNT, SUM, AVG, and MAX and MIN),

GROUP BY, HAVING and Creation and dropping of Views.

Experiment 10: Triggers

Work on Triggers. Creation of, insert trigger, delete trigger, update trigger. Practice triggers using the above database.

OBJECT ORIENTED PROGRAMMING

1. Learn the concepts of data, abstraction and encapsulation
2. Be able to write programs using classes and objects, packages.
3. Understand conceptually principles of Inheritance and Polymorphism and their use and program level implementation.
4. Learn exception and basic event handling mechanisms in a program
5. To learn typical object-oriented constructs of specific object oriented programming language

SYLLABUS

A. Theory **4 credits**

UNIT I. Basics: Introduction to Object Oriented Programming and its Basic Features, Basic Components of C++, Characteristics of Object-Oriented Language, Structure of a C++ Program, Flow Control Statements in C++, Functions - Scope of Variables, Inline Functions, Recursive Functions, Pointers to Functions, C++ Pointers, Arrays, Dynamic Memory Allocation and De-Allocation

UNIT II. Differences Between Object Oriented and Procedure Oriented Programming, Abstraction, Overview of Object-Oriented Programming Principles, Encapsulation, C++ Classes, Objects, User Defined Types, Constructors and Destructors, this Pointer, Friend Functions, Data Abstraction, Operator Overloading, Type Conversion

UNIT III. Class Inheritance, Base and Derived Classes, Virtual Base Class, Virtual Functions, Polymorphism, Static and Dynamic Bindings, Base and Derived Class Virtual Functions, Dynamic Binding through Virtual Functions, Pure Virtual Functions, Abstract Classes, Virtual Destructors

UNIT IV. Stream Classes Hierarchy, Stream I/O, File Streams, Overloading the Extraction and Insertion Operators, Error Handling during File Operations, Formatted I/O.

UNIT V. Exception Handling- Benefits of Exception Handling, Throwing an Exception, the Try Block, Catching an Exception, Exception Objects, Exception Specifications, Rethrowing an Exception, Uncaught Exceptions

TEXT BOOKS

- Problem solving with C++: The Object of Programming, Walter Savitch, 4th Edition, Pearson Education.
- C++: The Complete Reference, Herbert Schildt, 4th Edition

REFERENCE BOOKS

- Object Oriented Programming with C++, Sourav Sahay, 2nd Edition, Oxford
- The C++ Programming Language, B. Stroutstrup, 3rd Edition, Pearson Education
- Programming in C++, Ashok N Kamthane. Pearson 2nd Edition

B. Practicum **2 credits**

Students are required to understand the object-oriented concepts using C++. They are required to practice the concepts learnt in the theory . Some of the programs to be implemented are listed as follows:

Part A

1. Number of vowels and number of characters in a string.
2. Write a function called zeros maller () that is passed with two introduce arguments by reference and set the smaller of the number to zero. Write a man() program to access this function.
3. Demonstration of array of object.
4. Using this pointer to return a value (return by reference).
5. Demonstration of virtual function.
6. Demonstration of static function.
7. Accessing a particular record in a student's file.
8. Demonstration of operator overloading.

Part B

9. Write a program to create a database for students that contains Name, Enrolment no, Department, Programme using Constructors, destructors, input and output functions ; input and output for 10 people using different methods.

10. Create a class holding information of the salaries of all the family members (husband, wife, son, daughter). Using friend functions give the total salary of the family.

INTERNET TECHNOLOGIES

1. To understand the terms related to the Internet and how the Internet is changing the world.
2. To understand how computers are connected to the Internet and demonstrate the ability to use the World Wide Web.
3. Demonstrate an understanding of and the ability to use electronic mail and other internet based services
4. Understand the design principles of Web pages and how they are created
5. To develop an ability to create basic Web pages with HTML.

SYLLABUS **6 credits**

UNIT I. Introduction: Overview, Network of Networks, Intranet, Extranet and Internet. World Wide Web, Domain and Sub domain, Address Resolution, DNS, Telnet, FTP, HTTP. Review of TCP/IP: Features, Segment, Three-Way Handshaking, Flow Control, Error Control, Congestion control,

UNIT II. IP Datagram, IPv4 and IPv6. IP Subnetting and addressing: Classful and Classless Addressing, Subnetting. NAT, IP masquerading, IP tables. Internet Routing Protocol: Routing -Intra and Inter Domain Routing, Unicast and Multicast Routing, Broadcast. Electronic Mail: POP3, SMTP.

UNIT III. HTML: Introduction, Editors, Elements, Attributes, Heading, Paragraph. Formatting, Link, Head, Table, List, Block, Layout, CSS. Form, Iframe, Colors, Colorname, Colorvalue. Image Maps: map, area, attributes of image area. Extensible Markup Language (XML): Introduction, Tree, Syntax, Elements, Attributes, Validation, Viewing. XHTML in brief. CGI Scripts: Introduction, Environment Variable, GET and POST Methods.

UNIT IV. PERL: Introduction, Variable, Condition, Loop, Array, Implementing data structure, Hash, String, Regular Expression, File handling, I/O handling. JavaScript: Basics, Statements, comments, variable, comparison, condition, switch, loop, break. Object - string, array, Boolean, reg-ex. Function, Errors, Validation. Cookies: Definition of cookies, Create and Store a cookie with example. Java Applets: Container Class, Components, Applet Life Cycle, Update method; Parameter passing applet, Applications.

UNIT V. Client-Server programming In Java: Java Socket, Java RMI. Threats: Malicious code-viruses, Trojan horses, worms; eavesdropping, spoofing, modification, denial of service attacks. Network security techniques: Password and Authentication; VPN, IP Security, security in electronic transaction, Secure Socket Layer (SSL), Secure Shell (SSH). Firewall: Introduction, Packet filtering, Stateful, Application layer, Proxy.

UNIT VI. Internet Telephony: Introduction, VoIP. Multimedia Applications: Multimedia over IP: RSVP, RTP, RTCP and RTSP. Streaming media, Codec and Plugins, IPTV. mywbut.com Search Engine and Web Crawler: Definition, Meta data, Web Crawler, Indexing, Page rank, overview of SEO.

REFERENCE BOOKS

- Web Technology: A Developer's Perspective, N.P. Gopalan and J. Akilandeswari, PHI, Learning, Delhi, 2013.
- Internetworking Technologies, An Engineering Perspective, Rahul Banerjee, PHI Learning, Delhi, 2011.

ARTIFICIAL INTELLIGENCE

1. Explain what constitutes "Artificial" Intelligence and how to identify systems with Artificial Intelligence.
2. Identify problems that are amenable to solution by AI methods, and which AI methods may be suited to solving a given problem.
3. Formalise a given problem in the language/framework of different AI methods (e.g., as a search problem, as a constraint satisfaction problem, as a planning problem, etc).
4. Implement basic AI algorithms (e.g., standard search or constraint propagation algorithms).
5. Design and perform an empirical evaluation of different algorithms on a problem formalisation, and state the conclusions that the evaluation supports.
6. Explain the limitations of current Artificial Intelligence techniques.

SYLLABUS

A. Theory **4 credits**

UNIT I. Introduction to Artificial Intelligence: Definition of AI; Turing Test; Brief History of AI. Problem Solving and Search: Problem Formulation; Search Space; States vs. Nodes; Tree Search: Breadth-First, Uniform Cost, Depth-First, Depth-Limited, Iterative Deepening; Graph Search.

UNIT II. Informed Search: Greedy Search; A* Search; Heuristic Function; Admissibility and Consistency; Deriving Heuristics via Problem Relaxation. Local Search: Hill-Climbing; Simulated Annealing; Genetic Algorithms; Local Search in Continuous Spaces.

UNIT III. Playing Games: Game Tree; Utility Function; Optimal Strategies; Minimax Algorithm; Alpha-Beta Pruning; Games with an Element of Chance. Beyond Classical Search: Searching with Nondeterministic Actions; Searching with Partial Observations; Online Search Agents; Dealing with Unknown Environments.

UNIT IV. Knowledge Representation and Reasoning: Ontologies, Foundations of Knowledge Representation and Reasoning, Representing and Reasoning about Objects, Relations, Events, Actions, Time, and Space; Predicate Logic, Situation Calculus, Description Logics, Reasoning with Defaults, Reasoning about Knowledge, Sample Applications.

UNIT V. Representing and Reasoning with Uncertain Knowledge: Probability, Connection to Logic, Independence, Bayes Rule, Bayesian Networks, Probabilistic Inference, and Sample Applications.

UNIT VI. Planning: The STRIPS Language; Forward Planning; Backward Planning; Planning Heuristics; Partial-Order Planning; Planning using Propositional Logic; Planning vs. Scheduling.

UNIT VII. Constraint Satisfaction Problems (CSPs): Basic Definitions; Finite vs. Infinite vs. Continuous Domains; Constraint Graphs; Relationship With Propositional Satisfiability, Conjunctive Queries, Linear Integer Programming, and Diophantine Equations; NP-

Completeness of CSP; Extension to Quantified Constraint Satisfaction (QCSP). Constraint Satisfaction as a Search Problem; Backtracking Search; Variable and Value Ordering Heuristic; Degree Heuristic; Least-Constraining Value Heuristic; Forward Checking; Constraint Propagation; Dependency-Directed Backtracking;

TEXT BOOKS

- Elaine Rich, Kevin Knight, Shivashankar B Nair, Artificial Intelligence, Third Edition, McGraw Hill Edition.

REFERENCE BOOKS

- Russell Stuart Jonathan and Norvig Peter, Artificial Intelligence: A Modern Approach, 3rd Edition, Prentice Hall, 2010

B. Practicum **2 credits**

The students are expected to explore the foundational skills on AI techniques acquired in theory in solving problems and using sample data sets and various tools prepare themselves for careers in AI industry. The following is an indicative list of assignments for the semester. However students should be encouraged to take-up mini-project using the techniques and tools explored in the lab to understand the true potential

1. Using simple Hill-climbing compute an approximate solution to the travelling salesperson problem.

2. Using Naïve bayes method learn a text classifier using training data and using test set evaluate the quality of the classifier.

3. Implement gradient descent and backpropagation in Python.

4. Using Scikit learn for Logistic regression, Support Vector Machines, Building Neural Networks.

5. Using inbuilt TensorFlow functionality to build a Neural Network and train on MNIST Dataset for classification.

6. Installation of Prolog and practicing queries using Prolog.

COMPUTER GRAPHICS

1. Acquire familiarity with the concepts and relevant mathematics of computer graphics.
2. Ability to implement various algorithms to scan, convert the basic geometrical primitives, transformations, area filling, clipping.
3. Describe the importance of viewing and projections.
4. Ability to design basic graphics application programs.
5. Familiarize with fundamentals of animation and Virtual reality technologies
6. Be able to design applications that display graphic images to given specifications.
7. To understand a typical graphics pipeline.

SYLLABUS

A. Theory **4 credits**

UNIT I. Application Areas of Computer Graphics, Overview of Graphics Systems and Devices. Points and Lines, Line Drawing Algorithms, Mid-Point Circle and Ellipse Algorithms. Filled Area Primitives, Polygon Filling Algorithms. Curve Generation: Bezier and B-Spline Curves.

UNIT II. 2-D Geometrical Transforms: Translation, Scaling, Rotation, Reflection and Shear Transformations Composite Transforms, Transformations between Coordinate Systems. 2-D Viewing: The Viewing Pipeline, Viewing Coordinate Reference Frame, Window to Viewport Coordinate Transformation, Viewing Functions.

UNIT III. Line Clipping Algorithms- Cohen-Sutherland and Cyrus Beck Line Clipping Algorithms, Sutherland–Hodgeman Polygon Clipping Algorithm. 3-D Object Representation: Polygon Surfaces, Quadric Surfaces, Spline Representation

UNIT IV. 3-D Geometric Transformations: Translation, Rotation, Scaling, Reflection and Shear Transformations, Composite Transformations, 3-D Viewing: Viewing Pipeline, Viewing Coordinates, View Volume, General Projection Transforms and Clipping.

UNIT V. Visible Surface Detection Methods: Classification, Back-Face Detection, Depth-Buffer, Scanline, Depth Sorting, BSP-Tree Methods, Area Sub-Division and Octree Methods Illumination Models and Surface Rendering Methods: Basic Illumination Models, Polygon Rendering Methods Computer Animation: Design of Animation Sequence, General Computer Animation Functions Key Frame Animation, Animation Sequence, Motion Control Methods, Morphing, Warping (Only Mesh Warping)

UNIT VI. Virtual Reality : Basic Concepts, Classical Components of VR System, Types of VR Systems, Three Dimensional Position Trackers, Navigation and Manipulation Interfaces, Gesture Interfaces. Input Devices, Graphical Rendering Pipeline, Haptic Rendering Pipeline, Open GL Rendering Pipeline. Applications of Virtual Reality.

TEXTBOOKS

- Donald Hearn and M. Pauline Baker, "Computer Graphics with Open GL", Prentice Hall.
- R. K Maurya, "Computer Graphics with Virtual Reality", Wiley

REFERENCE BOOKS

- "Computer Graphics Principles & practice", Foley, Van Dam, Feiner and Hughes, Pearson Education.

B. Practicum **2 credits**

The students are required to create interactive graphics applications in C using graphics application programming interfaces and demonstrate geometrical transformations. The lab material includes implementation of line drawings, circle drawing, ellipse drawing as well as different geometrical transformations.

Experiment 1: Line Drawing Using DDA and Bresenham
Experiment 2: Circle Drawing Using Midpoint Algorithm .
Experiment 3: Ellipse Drawing Using Mipoint Algorithm.
Experiment 4: Performing the basic 2D transformations such as translation, Scaling, Rotation, shearing and reflection for a given 2D object.

MACHINE LEARNING

1. Differentiate between supervised, unsupervised machine learning approaches
2. Ability to choose appropriate machine learning algorithm for solving a problem
3. Design and adapt existing machine learning algorithms to suit applications
4. Understand the underlying mathematical relationships across various machine learning algorithms
5. Design and implement machine learning algorithms to real world applications

SYLLABUS **6 credits**

UNIT I. Introduction: Concept of Machine Learning, Applications of Machine Learning, Key elements of Machine Learning, Supervised vs. Unsupervised Learning, Statistical Learning: Bayesian Method, The Naive Bayes Classifier

UNIT II. Software's for Machine Learning and Linear Algebra Overview: Plotting of Data, Vectorization, Matrices and Vectors: Addition, Multiplication, Transpose and Inverse using Available Tool such as MATLAB.

UNIT III. Linear Regression: Prediction using Linear Regression, Gradient Descent, Linear Regression with one Variable, Linear Regression with Multiple Variables, Polynomial Regression, Feature Scaling/Selection.

UNIT IV. Logistic Regression: Classification using Logistic Regression, Logistic Regression vs. Linear Regression, Logistic Regression with one Variable and with Multiple Variables.

UNIT V. Regularization: Regularization and its Utility: The problem of Overfitting, Application of Regularization in Linear and Logistic Regression, Regularization and Bias/Variance.

UNIT VI. Neural Networks: Introduction, Model Representation, Gradient Descent vs. Perceptron Training, Stochastic Gradient Descent, Multilayer Perceptrons, Multiclass Representation, Back Propagation Algorithm.

TEXT BOOKS

- Ethem Alpaydin, "Introduction to Machine Learning" 2nd Edition, The MIT Press, 2009.
- Tom M. Mitchell, "Machine Learning", First Edition by Tata McGraw-Hill Education, 2013.
- Christopher M. Bishop, "Pattern Recognition and Machine Learning" by Springer, 2007.
- Mevin P. Murphy, "Machine Learning: A Probabilistic Perspective" by The MIT Press, 2012.

IMAGE PROCESSING

1. To familiarize the students with the image fundamentals and mathematical transforms necessary for image processing.
2. To make the students understand the image enhancement techniques
3. To make the students understand the image restoration and reconstruction procedures.
4. To familiarize the students with the image segmentation procedures.

SYLLABUS **6 credits**

UNIT I Digital Image Fundamentals: Elements of Visual Perception, Light, Brightness Adaption and Discrimination, Image Sensing and Acquisition, Image Sampling and Quantization, Pixels, Some Basic Relationships between Pixels, Coordinate Conventions, Imaging Geometry, Perspective Projection, Linear and Nonlinear Operations

UNIT II Image Enhancement in the Spatial Domain: Intensity transformations, Contrast Stretching, Histogram Equalization, Correlation and Convolution, Basics of Spatial Filtering, Smoothing Filters, Sharpening Filters, Gradient and Laplacian.

UNIT III Filtering in the Frequency domain: Hotelling Transform, Fourier Transforms and properties, FFT (Decimation in Frequency and Decimation in Time Techniques), Convolution, Correlation, 2-D sampling, Discrete Cosine Transform, Frequency domain filtering.

UNIT IV Image Restoration and Reconstruction: Basic Framework, Interactive Restoration, Image deformation and geometric transformations, image morphing, Restoration techniques, Noise characterization, Noise restoration filters, Adaptive filters, Linear, Position invariant degradations, Estimation of Degradation functions, Restoration from projections.

UNIT V Color Image Processing, Color Fundamentals, Color Models, Pseudocolor Image Processing, Basics of Full-Color Image Processing, Color Transformations, Smoothing and Sharpening, Color Segmentation. Morphological Image Processing, Dilation and Erosion, Opening and Closing., Extensions to Gray -Scale Images.

UNIT VI Image Segmentation: Detection of Discontinuities, Edge Linking and Boundary Detection, Thresholding, Region-Based Segmentation, Segmentation by Morphological Watersheds.

TEXT BOOKS

- Digital Image Processing, Rafael C. Gonzalez and Richard E. Woods, 4th Edition, Prentice Hall.

REFERENCE BOOKS

- Anil K. Jain, Fundamentals of Digital Image Processing, Prentice Hall.
- Stan Birchfield, Image Processing and Analysis, Cengage Learning.

DATA ANALYTICS

1. This course prepares students to gather, describe, and analyze data, and use advanced statistical tools to support decision making.
2. To gather sufficient relevant data, conduct data analytics using scientific methods, and understand appropriate connections between quantitative analysis and real-world problems.
3. Understand the exact scopes and possible limitations of each method to provide constructive guidance in decision making.
4. To Use advanced techniques to conduct thorough and insightful analysis, and interpret the results correctly with detailed and useful information.
5. To make better decisions by using advanced techniques in data analytics.

SYLLABUS **6 credits**

UNIT I. Data Definitions and Analysis Techniques: Elements, Variables, and Data Categorization, Levels of Measurement, Data Management and Indexing

UNIT II. Descriptive Statistics: Measures of Central Tendency, Measures of Location of Dispersions, Error Estimation and Presentation (Standard Deviation, Variance), Introduction to Probability

UNIT III. Basic Analysis Techniques: Statistical Hypothesis Generation and Testing, Chi-Square Test, T-Test, Analysis of Variance, Correlation Analysis, Maximum Likelihood Test

UNIT IV. Data Analysis Techniques-I: Regression Analysis, Classification Techniques, Clustering Techniques (K-Means, K-Nearest Neighborhood) UNIT

V. Data Analysis Techniques-II: Association Rules Analysis, Decision Tree

UNIT VI. Introduction to R Programming: Introduction to R Software Tool, Statistical Computations using R (Mean, Standard Deviation, Variance, Regression, Correlation etc.)

UNIT VII. Practice and Analysis with R and Python Programming, Sensitivity Analysis

REFERENCE BOOKS

- Probability and statistics for Engineers and Scientists (9 Edn.), Ronald E Walppole, Raymond H Myres, Sharon L. Myres and Leying Ye, Prentice Hall Inc
- The Elements of Statistical Learning, Data Mining, Inference, and Prediction (2nd Edn.) Travor Hastie Robert Tibshirani Jerome Friedman, Springer, 2014

- Software for Data Analysis: Programming with R (Statistics and Computing), John M. Chambers, Springer

COMPUTER ETHICS

1. The student will be able to describe and distinguish between the various ethical theories which can be used to form the basis of solutions to moral dilemmas in computing.
2. Identify traditional and current Issues related to Computers, Information Systems, Ethics, Society and Human Values;
3. The student will be able to identify and define the components of a structured plan for solving ethical problems and, in the process, will be able to understand the basis for her/his own ethical system.
4. Given several examples of professional codes of ethics related to computing, the student will be able to compare and contrast these examples, discussing their commonalties, differences, and implications.
5. Develop skills of critical analysis and applying ethical principles to situations and dialectical thinking

SYLLABUS **6 credits**

UNIT I. The Need for Computer Ethics Training and Historical Milestones

UNIT II. Defining the Field of Computer Ethics, Computer ethics codes, Sample Topics in Computer Ethics

i. Computer crime and computer security
ii. Software theft and intellectual property rights
iii. Computer hacking and the creation of viruses
iv. Computer and information system failure
v. Invasion of privacy. Privacy in the Workplace and on the Internet
vi. Social implications of artificial intelligence and expert systems
vii. The information technology salesman issues

UNIT III. Transparency and Virtual Ethics, Free Speech, Democracy, Information Access

UNIT IV. Developing the Ethical Analysis Skills and Professional Values, Privacy, Accountability, Government Surveillance

UNIT V. Boundaries of Trust, Trust Management, Wikipedia, Virtual Trust, Plagiarism in Online Environment, Intellectual Property, Net neutrality

REFERENCE BOOKS

- Deborah, J, Nissenbaun, H, Computing, Ethics & Social Values, Englewwod Cliffs, New Jersey, Prentice Hall, 1995.
- Spinello, R, Tavani, H, T, Readings in Cyberethics, Sudbury, MA, Jones and Bartlett Publishers, 2001.
- Bynum, T, W; Rogerson, S, Computer Ethics and Professional Responsibility, Blackwell, 2004

SYSTEM SECURITY

1. Develop an understanding of information assurance as practiced in computer operating systems, distributed systems, networks and representative applications.
2. Gain familiarity with prevalent network and distributed system attacks, defenses against them, and forensics to investigate the aftermath.
3. Develop a basic understanding of cryptography, how it has evolved, and some key encryption techniques used today.
4. Develop an understanding of security policies (such as authentication, integrity and confidentiality), as well as protocols to implement such policies in the form of message exchanges.

SYLLABUS **6 Credits**

UNIT 1. Cryptographic Tools- Confidentiality with Symmetric Encryption, Message Authentication and Hash Functions, Public-Key Encryption, Digital Signatures and Key Management, Random and Pseudorandom Numbers, Practical Application: Encryption of Stored Data

UNIT 2. User Authentication- Means of Authentication, Password-Based Authentication, Token-Based Authentication, Biometric Authentication, Remote User Authentication, Security Issues for User Authentication, Practical Application: An Iris Biometric System, Case Study: Security Problems for ATM Systems

UNIT 3. Access Control- Access Control Principles, Subjects, Objects, and Access Rights, Discretionary Access Control, Example: UNIX File Access Control, Role-Based Access Control, Case Study: RBAC System for a Bank

UNIT 4. Database Security-The Need for Database Security, Database Management Systems, Relational Databases, Database Access Control, Inference, Statistical Databases, Database Encryption, Cloud Security

UNIT 5. Malicious Software-Types of Malicious Software (Malware), Propagation–Infected Content–Viruses, Propagation–Vulnerability Exploit–Worms, Propagation–Social Engineering–SPAM E-mail, Trojans, Payload–System Corruption, Payload–Attack Agent–Zombie, Bots, Payload–Information Theft–Keyloggers, Phishing, Spyware, Payload–Stealthing–Backdoors, Rootkits,, Countermeasures

UNIT 6. Denial-of-Service Attacks- Denial-of-Service Attacks, Flooding Attacks, Distributed Denial-of-Service Attacks, Application-Based Bandwidth Attacks, Reflector and Amplifier Attacks, Defenses Against Denial-of-Service Attacks, Responding to a Denial-of-Service Attack.

TEXT BOOKS

- M. Stamp, “Information Security: Principles and Practice,” 2 st Edition, Wiley, ISBN: 0470626399, 2011.
- M. E. Whitman and H. J. Mattord, “Principles of Information Security,” 4 st Edition, Course Technology, ISBN: 1111138214, 2011.
- M. Bishop, “Computer Security: Art and Science,” Addison Wesley, ISBN: 0-201-44099-7, 2002.
- G. McGraw, “Software Security: Building Security In,” Addison Wesley, ISBN: 0321356705, 2006.

HUMAN COMPUTER INTERFACE

1. Provide an overview of the concepts relating to the design of human-computer interfaces in ways making computer-based systems comprehensive, friendly and usable.
2. Understand the theoretical dimensions of human factors involved in the acceptance of computer interfaces.
3. Understand the important aspects of implementation of human-computer interfaces.
4. Identify the various tools and techniques for interface analysis, design, and evaluation.
5. Identify the impact of usable interfaces in the acceptance and performance utilization of information systems.

SYLLABUS **6 credits**

UNIT I. Introduction: Historical Evolution of HCI, Interactive System Design: Concept of Usability- Definition and Elaboration, HCI and Software Engineering, GUI Design and Aesthetics, Prototyping Techniques

UNIT II. Model-Based Design and Evaluation: Basic Idea, Introduction to Different Types of Models, GOMS Family of Models (KLM And CMN-GOMS), Fitts' Law and Hickhyman's Law,

UNIT III. General Development Guidelines and Principles: Shneiderman's Eight Golden Rules, Norman's Seven Principles, Norman's Model of Interaction, Nielsen's Ten Heuristics with Example of its use, Contextual Inquiry

UNIT IV. Dialog Design: Introduction to Formalism in Dialog Design, Design using FSM (Finite State Machines), State Charts and (Classical) Petri Nets in Dialog Design

UNIT V. Task Modeling and Analysis: Hierarchical Task Analysis (HTA), Engineering Task Models and Concur Task Tree (CTT)

UNIT VI. Object Oriented Modelling: Object Oriented Principles, Definition of Class and Object and their Interactions, Object Oriented Modelling for User Interface Design, Case Study Related to Mobile Application Development

REFERENCE BOOKS

- Dix A., Finlay J.,Abowd G. D. and Beale R. Human Computer Interaction, 3 rd edition, Pearson Education, 2005.
- Preece J., Rogers Y.,Sharp H.,Baniyon D., Holland S. and Carey T. Human Computer Interaction,Addison-Wesley, 1994.
- B.Shneiderman; Designing the User Interface,Addison Wesley 2000 (Indian Reprint).

MODELLING AND SIMULATIONS

1. Characterise systems in terms of their essential elements, purpose, parameters, constraints, performance requirements, sub-systems, interconnections and environmental context.
2. Understand the technical underpinning of modern computer simulation software.
3. System problem modelling and solving through the relationship between theoretical, mathematical, and computational modelling for predicting and optimizing performance and objective.
4. Mathematical modelling real world situations related to information systems development, prediction and evaluation of outcomes against design criteria.
5. Develop solutions and extract results from the information generated in the context of the information systems
6. Interpret the model and apply the results to resolve critical issues in a real world environment.
7. Develop different models to suit special characteristics of the system being modelled.

SYLLABUS **6 credits**

UNIT 1. Systems and environment: Concept of model and model building, model classification and representation, Use of simulation as a tool, steps in simulation study.

UNIT 2. Continuous-time and Discrete-time systems: Laplace transform, transfer functions, statespace models, order of systems, z-transform, feedback systems, stability, observability, controllability. Statistical Models in Simulation: Common discrete and continuous distributions, Poisson process, empirical distributions

UNIT 3. Random Numbers: Properties of random numbers, generation of pseudo random numbers, techniques of random number generation, tests for randomness, random variate generation using inverse transformation, direct transformation, convolution method, acceptance-rejection

UNIT 4. Design and Analysis of simulation experiments: Data collection, identifying distributions with data, parameter estimation, goodness of fit tests, selecting input models without data, multivariate an time series input models, verification and validation of models, static and dynamic simulation output analysis, steady-state simulation, terminating simulation, confidence interval estimation, Output analysis for steady state simulation, variance reduction techniques

UNIT 5. Queuing Models: Characteristics of queuing systems, notation, transient and steady state behavior, performance, network of queues

UNIT 6. Large Scale systems: Model reduction, hierarchical control, decentralized control, structural properties of large-scale systems

REFERENCE BOOKS

- Shailendra Jain, Modeling and Simulation using MATLAB - Simulink, 2ed, Kindle edition

THEORY OF COMPUTATION

1. To provide a formal connection between algorithmic problem solving and the theory of languages and automata and develop them into a mathematical (abstract) view towards algorithmic design and in general computation itself.

2. The course should in addition clarify the practical view towards the applications of these ideas in the engineering part as well.

3. Become proficient in key topics of theory of computation, and to have the opportunity to explore the current topics in this area

PREREQUISITE

Students should have a background in discrete mathematics, data structures, and programming languages.

SYLLABUS

A THEORY **4 Credits**

UNIT I. Automata: Introduction to Formal Proof, Additional Forms of Proof, Inductive Proofs, Finite Automata (FA), Deterministic Finite Automata (DFA), Non-Deterministic Finite Automata (NFA), Finite Automata with Epsilon Transitions

UNIT II. Regular Expressions and Languages: Regular Expression, FA and Regular Expressions, Proving Languages not to be Regular, Closure Properties of Regular Languages, Equivalence and Minimization of Automata

UNIT III. Context Free Grammars and Languages: Context Free Grammar (CFG), Parse Trees , Ambiguity in Grammars and Languages, Definition of The Pushdown Automata, Languages of a Pushdown Automata, Equivalence of Pushdown Automata and CFG Deterministic Pushdown Automata.

UNIT IV. Properties of Context Free Languages: Normal Forms for CFG, Pumping Lemma for CFL, Closure Properties of CFL, Turing Machines, Programming Techniques for TM, Variations of TM, Non Universal TM, Universal TM.

UNIT V. Undecidability: A Language that is not Recursively Enumerable (RE), an Undecidable Problem that is RE, Undecidable Problems about Turing Machine, Post's Correspondence Problem, The Classes P and NP.

REFERENCE BOOKS

- J.E. Hopcroft, R. Motwani and J.D. Ullman, "Introduction to Automata Theory, Languages and Computations", second Edition, Pearson Education, 2007.
- H.R. Lewis and C.H. Papadimitriou, "Elements of the theory of Computation", Second Edition, Pearson Education, 2003.

- Thomas A. Sudkamp," An Introduction to the Theory of Computer Science, Languages and Machines", Third Edition, Pearson Education., 2007.
- J. Martin, "Introduction to Languages and the Theory of computation, Third Edition, Tata Mc Graw Hill, 2007.

B. Practicum **2 credits**

The students are expected to understand the Hierarchy of formal languages with reference to their varying degrees of complexity in recognising them. Programs can be designed after designing suitable automata to recognize the following formal languages. Given an input the recognizer shall output a Yes/No answer depending on whether the string is part of the language or not.

1. Language of Binary strings which ends with the pattern 101.
2. Language of Binary strings such that the third symbol from the end is a Zero
3. Language of parenthesised expressions with matching left and right parenthesis
4. Language of Binary strings with equal number of Zeros and Ones
5. Language generated by the grammar $\{a^n b^n c^n \mid n^3 1\}$
6. Language $\{ a^p \mid p \text{ is prime}\}$

DATA MINING

1. Demonstrate advanced knowledge of data mining concepts and techniques.
2. Apply the techniques of clustering, classification, association finding, feature selection and visualisation on real world data
3. Determine whether a real world problem has a data mining solution
4. Apply data mining software and toolkits in a range of applications
5. Set up a data mining process for an application, including data preparation, modelling and evaluation
6. Demonstrate knowledge of the ethical considerations involved in data mining.

SYLLABUS **6 credits**

UNIT I. Introduction to Data Mining, Understanding Data, Relations to Database, Statistics, Machine Learning

UNIT II. Association Rule Mining, Level-wise Method, FP-Tree Method, Other Variants

UNIT III. Classification, Decision Tree Algorithm, CART, PUBLIC, Pruning Classification Tree

UNIT IV. Clustering Techniques, Clustering of Numeric Data, of Ordinal Data, Efficiency of Clustering, Consensus Clustering, Spectral Clustering

UNIT V. Rough Set Theory and its Application to Data Mining

UNIT VI. ROC Analysis

UNIT VII. Data Mining Trends, Big Data, Data Analytics

TEXT BOOKS

- Data Mining Techniques (4e) Universities Press Arun K Pujari

CLOUD COMPUTING

1. Analyze the trade-offs between deploying applications in the cloud and over the local infrastructure.
2. Compare the advantages and disadvantages of various cloud computing platforms.
3. Deploy applications over commercial cloud computing infrastructures such as Amazon Web Services, Windows Azure, and Google AppEngine.
4. Program data intensive parallel applications in the cloud.
5. Analyze the performance, scalability, and availability of the underlying cloud technologies and software.
6. Identify security and privacy issues in cloud computing.
7. Explain recent research results in cloud computing and identify their pros and cons.
8. Solve a real-world problem using cloud computing through group collaboration.

SYLLABUS

A. **Theory** **4 Credits**

Unit I. Introduction to cloud computing

Definition, characteristics, components, Cloud service provider, the role of networks in Cloud computing, Cloud deployment models- private, public & hybrid, Cloud service models, multitenancy, Cloud economics and benefits, Cloud computing platforms - IaaS: Amazon EC2, PaaS: Google App Engine, Microsoft Azure, SaaS.

Unit II. Virtualization

Virtualization concepts , Server virtualization, Storage virtualization, Storage services, Network virtualization, Service virtualization, Virtualization management, Virtualization technologies and architectures, virtual machine, Measurement and profiling of virtualized applications. Hypervisors: KVM, Xen, VMware hypervisors and their features.

Unit III. Data in cloud computing

Relational databases, Cloud file systems: GFS and HDFS, BigTable, HBase and Dynamo. MapReduce and extensions: Parallel computing, the map-Reduce model, Parallel efficiency of MapReduce, Relational operations using Map-Reduce, Enterprise batch processing using MapReduce.

Unit IV. Cloud security

Cloud security fundamentals, Vulnerability assessment tool for cloud, Privacy and Security in cloud. Cloud computing security architecture: General Issues, Trusted Cloud computing, Secure Execution Environments and Communications, Micro - architectures; Identity Management and Access control, Autonomic security, Security challenges : Virtualization security management - virtual threats, VM Security Recommendations, VM - Specific Security techniques, Secure Execution Environments and Communications in cloud.

Unit V. Issues in cloud computing

Implementing real time application over cloud platform, Issues in Inter-cloud environments, QOS Issues in Cloud, Dependability, data migration, streaming in Cloud. Quality of Service (QoS) monitoring in a Cloud computing environment. Cloud Middleware. Mobile Cloud Computing. Inter Cloud issues. A grid of clouds, Sky computing, load balancing, resource optimization, resource dynamic reconfiguration, Monitoring in Cloud

.

TEXT BOOK:

1. Enterprise Cloud Computing by Gautam Shroff,Cambridge publication

REFERENCE BOOK:

1. Cloud Security by Ronald Krutz and Russell Dean Vines, Wiley-India
2.. Dr. Kumar Saurabh,"Cloud Computing", Wiley Publication

B. **Practicum** **2 Credits**

The students shall explore development of web applications in cloud. Practically Design and develop processes involved in creating a cloud based application and programming using Hadoop

Indicative List of Experiments

1. Install Virtualbox/VMware Workstation with different flavours of linux or windows OS with virtualization support
2. Install a C compiler in the virtual machine created using virtual box and execute Simple Programs
3. Install Google App Engine. Create hello world app and other simple web applications using python/java.
4. Simulate a cloud scenario using CloudSim and run a scheduling algorithm that is not present in CloudSim.
5. Experiment a procedure to transfer the files from one virtual machine to another virtual machine.
6. Experiment a procedure to launch virtual machine using trystack (Online Openstack Demo Version)
7. Install Hadoop single node cluster and run simple applications like word count.

INTERNET OF THINGS

1. To learn the concepts of Sensors, Wireless Network and Internet
2. To learn and implement use of Devices in IoT technology.
3. To learn the different IoT Technologies like Micro-controller, Wireless communication like Blue Tooth, GPRS, Wi-Fi and Storage and embedded systems
4. To understand how to program on embedded and mobile platforms including different Microcontrollers like ESP8266, Raspberry Pi, Arduino and Android programming
5. To understand how to make sensor data available on the Internet (data acquisition) and understand how to analyze and visualize sensor data
6. To understand, analysis and evaluate different protocols used in IoT.
7. To learn basic python programming for IoT applications
8. To learn and design different applications in IoT.
9. To design, develop and test different prototypes in IoT.

SYLLABUS **6 credits**

UNIT I. (Introduction to IoT, Sensors and Actuators) Introduction to IoT: Definition, Characteristics, Applications, Evolution, Enablers, Connectivity Layers, Addressing, Networking and Connectivity Issues, Network Configurations, Multi-Homing, Sensing: Sensors and Transducers, Classification, Different Types of Sensors, Errors, Actuation: Basics, Actuator Types- Electrical, Mechanical Soft Actuators

UNIT II. (Introduction to Networking, Communication Protocols and Machine-to-Machine Communication) Basics of Networking, Communication Protocols, Sensor Network, Machine to Machine Communication (IoT Components, Inter-Dependencies, SoA, Gateways, Comparison Between IoT & Web, Difference Protocols, Complexity of Networks, Wireless Networks, Scalability, Protocol Classification, MQTT & SMQTT, IEEE 802.15.4, Zigbee)

UNIT III. (Arduino Programming) Interoperability in IoT, Introduction To Arduino Programming, Integration Of Sensors And Actuators With Arduino

UNIT IV. (Python Programming and Raspberry Pi) Introduction to Python Programming, Introduction to Raspberry Pi, Implementation of IoT with Raspberry Pi, Implementation of IoT with Raspberry Pi

UNIT V. (Data Analytics and Cloud Computing) Data Handling and Analytics, Cloud Computing Fundamentals, Cloud Computing Service Model, Cloud Computing Service Management and Security, Sensor-Cloud Architecture, View and Dataflow

UNIT VI. (FOG Computing and Case Studies) FOG Computing: Introduction, Architecture, Need, Applications and Challenges

UNIT VII. Industrial IoT, Case Studies: Agriculture, Healthcare, Activity Monitoring

REFERENCE BOOKS

- The Internet of Things: Enabling Technologies, Platforms, and Use Cases", by Pethuru Raj and Anupama C. Raman (CRC Press).
- Internet of Things: A Hands-on Approach", by A Bahga and Vijay Madisetti (Universities Press)

MATLAB PROGRAMMING

1. Understand the fundaments of procedural and functional programming;
2. Understand Matlab data types and structures;
3. Be able to set up simple real-life numerical problems such that they can be solved and visualized using basic codes in Matlab;
4. Be ready to use advanced coding in Matlab in their subsequent studies

SYLLABUS **4 Credits**

UNIT 1. Introduction to MATLAB Programming- Basics of MATLAB programming, Array operations in MATLAB, Loops and execution control, Working with files: Scripts and Functions, Plotting and program output

UNIT 2. Approximations and Errors- Defining errors and precision in numerical methods, Truncation and round-off errors, Error propagation, Global and local truncation errors

UNIT 3. Linear Equations- Linear algebra in MATLAB, Gauss Elimination, LU decomposition and partial pivoting, Iterative methods: Gauss Siedel Method

UNIT 4. Regression and Interpolation- Introduction, Linear least squares regression(including lsqcurvefit function), Functional and nonlinear regression (including *lsqnonlin* function), Interpolation in MATLAB using spline and *pchip*

UNIT 5. Nonlinear Equations- Nonlinear equations in single variable, MATLAB function fzero in single variable, Fixed-point iteration in single variable, Newton-Raphson in single variable, MATLAB function *fsolve* in single and multiple variables, Newton-Raphson in multiple variables

TEXT BOOKS

1. Fausett L.V.(2007) Applied NumericalAnalysis Using MATLAB, 2nd Ed.,Pearson Education
2. Essential MATLAB for Engineers and Scientists, 6th Edition, Brian Hahn; Daniel T. Valentine, Academic Press, Web ISBN-13: 978-0-12-805271-6,

PROGRAMMING IN JAVA

1. Knowledge of the structure and model of the Java programming language,
2. Use the Java programming language for various programming technologies
3. Develop software in the Java programming language,
4. Evaluate user requirements for software functionality required to decide whether the Java programming language can meet user requirements

SYLLABUS **4 credits**

A. Theory

UNIT I. Introduction: Java Essentials, Its characteristics, Execution and Compilation, Data types, Variables, Control Statements, Standard Input/ Output.

UNIT II. Constructors, Object Oriented Concepts: Encaptulation, Abstraction, Inheritance, Polymorphisms, JAVA Packages.

UNIT III. Exception Handling, Wrapper Classes, Autoboxing, Multi-thread Programming.

UNIT IV. Applets, Event Handling, AWT, Database Handling using JDBC.

TEXT BOOKS

- E Balaguruswamy, Programming with JAVA, A Primer (5e), Kindle Edition

REFERENCE BOOKS

- Bruce Eckel, Thinking in Java (4e)
- Herbert Schildt, Java: The Complete Reference (9e)
- Y. Daniel Liang, Introduction to Java Programming (10e)
- Paul Deitel, Harvey Deitel, Java: How To Program (10e)
- Cay S. Horsttnann, Core Java Volume I –Fundamentals (10e)

B. Practicum

Students are required to implement object-oriented paradigm using JAVA. Below are the list of some of the experiments.

Part A

1. Program on strings: Check the equality of two strings, Reverse a string.
2. Program using loops: to find the sum of digits of a given number, display a multiplication table, display all prime numbers between 1 to 1000.
3. Program to demonstrate all math class functions.

Part B

4. Program on files : to copy a file to another file using Java to package classes.
5. Program to demonstrate method over-riding and overloading
6. Programs on inheritances.
7. Multi-threaded programming.

PYTHON PROGRAMMING

1. Develop and Execute simple Python programs.
2. Structure a Python program into functions.
3. Using Python lists, tuples to represent compound data
4. Develop Python Programs for file processing

SYLLABUS

A Theory **4 Credits**

UNIT I. Introduction to Python, Python, Features of Python, Execution of a Python, Program, Writing Our First Python Program, Data types in Python. Python Interpreter and Interactive Mode; Values and Types: int, float, boolean, string, and list; Variables, Expressions, Statements, Tuple Assignment, Precedence of Operators, Comments; Modules and Functions, Function Definition and use, Flow of Execution, Parameters and Arguments

UNIT II. Operators in Python, Input and Output, Control Statements. Boolean Values and operators, Conditional (if), Alternative (if-else), Chained Conditional (if-elif-else); Iteration: state, while, for, break, continue, pass; Fruitful Functions: Return Values, Parameters, Local and Global Scope, Function Composition, Recursion

UNIT III. Arrays in Python, Strings and Characters. Strings: String Slices, Immutability, String Functions and Methods, String Module; Lists as Arrays. Illustrative Programs: Square Root, gcd, Exponentiation, Sum an Array of Numbers, Linear Search, Binary Search.

UNIT IV. Functions, Lists and Tuples. List Operations, List Slices, List Methods, List Loop, Mutability, Aliasing, Cloning Lists, List Parameters; Tuples: Tuple Assignment, Tuple as Return Value; Dictionaries: Operations and Methods; Advanced List Processing - List Comprehension; Illustrative Programs: Selection Sort, Insertion Sort, Merge sort, Histogram.

UNIT V. Files and Exception: Text Files, Reading and Writing Files, Format Operator; Command Line Arguments, Errors and Exceptions, Handling Exceptions, Modules, Packages; Illustrative Programs: Word Count, Copy File.

TEXT BOOKS

- Mark Lutz, Learning Python
- Tony Gaddis, Starting Out With Python
- Kenneth A. Lambert, Fundamentals of Python
- James Payne, Beginning Python using Python 2.6 and Python 3

B. Practicum **2 Credits**

The students are required to verify their ablility to use core programming basics and program design with functions using Python programming language. The teacher shall programs to strengthen the practical expertise of the students. The following is an indicative list of programs that can be practised

1. Write a program to demonstrate different number data types in Python.

2. Write a program to perform different Arithmetic Operations on numbers in Python.

3. Write a program to create, concatenate and print a string and accessing sub-string from a given string.

4. Write a python script to print the current date in the following format "Fri Oct 11 02:26:23 IST 2019"

5. Write a program to create, append, and remove lists in python.

6. Write a program to demonstrate working with tuples in python.

7. Write a program to demonstrate working with dictionaries in python.

8. Write a python program to find largest of three numbers.

9. Write a Python program to construct the following pattern, using a nested for loop

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

10. Write a Python script that prints prime numbers less than 20.

11. Write a python program to define a module to find Fibonacci Numbers and import the module to another program.

12. Write a python program to define a module and import a specific function in that module to another program.

13. Write a program that inputs a text file. The program should print all of the unique words in the file in alphabetical order.

14. Write a Python class to convert an integer to a roman numeral.

15. Write a Python class to reverse a string word by word.

MOBILE APPLICATION DEVELOPMENT

1. To understand Android platform and its architecture.
2. To learn about mobile devices types and different modern mobile operating systems.
3. To learn activity creation and Android User Interface designing.
4. To learn basics of Intent, Broadcast and Internet services.
5. To learn about different wireless mobile data transmission standards.
6. To understand and learn how to integrate basic phone features, multimedia, camera and Location based services in Android Application.
7. To learn about different systems for mobile application development, deployment and distribution in Mobile market place (Android, IoS).
8. To understand and carry out functional test strategies for mobile applications.

SYLLABUS **4 credits**

UNIT I. (Introduction) What is Android, Android Versions and its Feature Set, Various Android Devices on the Market, Android Market Application Store, Android Development Environment System Requirements, Android SDK, Installing Java, and ADT bundle - Eclipse Integrated Development Environment (IDE), Creating Android Virtual Devices (AVDs)

UNIT II. (Android Architecture Overview and Application) Android Software Stack, The Linux Kernel, Android Runtime - Dalvik Virtual Machine, Android Runtime – Core Libraries, Dalvik VM Specific Libraries, Java Interoperability Libraries, Android Libraries, Application Framework, Creating a New Android Project ,Defining the Project Name and SDK Settings, Project Configuration Settings, Configuring the Launcher Icon, Creating an Activity, Running the Application in the AVD, Stopping a Running Application, Modifying the Example Application, Reviewing the Layout and Resource Files,

UNIT III. (Android Software Development Platform and Framework) Understanding Java SE and the Dalvik Virtual Machine, The Directory Structure of an Android Project, Common Default Resources Folders, The Values Folder, Leveraging Android XML, Screen Sizes , Launching Mobile Application: The AndroidManifest.xml File, Android Application Components, Android Activities: Defining the UI, Android Service s: Processing in the Background, Broadcast Receivers: Announcements and Notifications Content Providers: Data Management, Android Intent Objects: Messaging for Components, Android Manifest XML: Declaring Your Components

UNIT IV. (Understanding Android User Interfaces, Views and Layouts) Designing for Different Android Devices, Views and View Groups, Android Layout Managers, The View Hierarchy, Designing an Android User Interface using the Graphical Layout Tool Displaying Text with TextView, Retrieving Data from Users, Using Buttons, Check Boxes and Radio Groups, Getting Dates and Times from Users, Using Indicators to Display Data to Users, Adjusting Progress with Seek Bar, Working with Menus using views, Gallery, Image Switcher, Grid View, and Image View views to display images, Creating Animation

UNIT V. (Databases, Intents, Location-based Services) Saving and Loading Files, SQLite Databases, Android Database Design, Exposing Access to a Data Source through a Content

Provider, Content Provider Registration, Native Content Providers Intents and Intent Filters: Intent Overview, Implicit Intents, Creating the Implicit Intent Example Project, Explicit Intents, Creating the Explicit Intent Example Application, Intents with Activities, Intents with Broadcast Receivers

UNIT VI. Sending SMS Messages Programmatically, Getting Feedback after Sending the Message Sending SMS Messages Using Intent Receiving, sending email, Introduction to location-based service, configuring the Android Emulator for Location-Based Services, Geocoding and Map-Based Activities Multimedia: Audio, Video, Camera: Playing Audio and Video, Recording Audio and Video, Using the Camera to Take and Process Pictures

REFERENCE BOOKS

- Android Programming Unleashed (1st Edition) by Harwani.
- Beginning Mobile Application Development in the Cloud (2011), Richard Rodger.

WEB PROGRAMMING

1. To understand basics of the Internet and World Wide Web
2. To acquire knowledge and skills for creation of web site considering both client and server-side programming
3. To learn basic skill to develop responsive web applications
4. To understand different web extensions and web services standards
5. To understand basic concepts of Search Engine Basics.
6. To learn Web Service Essentials.
7. To learn Rich Internet Application Technologies.
8. To understand and get acquainted with Web Analytics 2.0

SYLLABUS **4 credits**

UNIT I. (Introduction to World Wide Web) -Internet Standards, Introduction to WWW and WWW Architecture, Internet Protocols, Overview of HTTP, HTTP request – response, Generations of dynamic web pages

UNIT II. (User Interface Design) Introduction to HTML and HTML5, TML Tags, Formatting and Fonts, Commenting Code, Anchors, Backgrounds, Images, Hyperlinks, Lists, Tables, Frames, HTML Forms. The need for CSS, Introduction to CSS, Basic syntax and structure, Inline Styles, Embedding Style Sheets, Linking External Style, Backgrounds, Manipulating Text, Margins and Padding, Positioning using CSS.

UNIT III. (Java Programming) Java Script, Introduction, Core features, Data types and Variables, Operators, Expressions, Functions, Objects, Array, Date and Math related Objects. JAVA Networking classes, TCP/IP Protocol Suite, File Transfer Protocol (FTP), Java Environment |Setup for Web Applications, JavaBean, Application Builder Tool, Bean Developer Kit (BDK), The Java Beans API, Introduction to EJB

UNIT IV. (Database) Database basics, SQL, MySQL, PostgreSQL, JDBC API, Driver Types, Two-tier and Three-tier Models, Connection Overview, Transactions, Driver Manager Overview, Statement Overview, Result Set Overview, Types of Result Sets, Concurrency Types, Prepared Statement Overview

UNIT V. (Java Applet and JSP) Java Web Programs and Applets, Web Application, Servlet, Servlet Life Cycle, Servlet Programming, Introduction to JSP, Life Cycle of a JSP Page, Translation and Compilation, Creating Static Content, Response and Page Encoding, Creating Dynamic Content, Using Objects within JSP Pages, JSP Programming

UNIT VI. (Dot Net Framework) Introduction to Dot Net, Dot Net framework and its architecture, CLR, Assembly, Components of Assembly, DLL hell and Assembly Versioning, Overview to C#, Introduction to ASP.net, Asp.net Programming

REFERENCE BOOKS

- J2EE: The complete Reference by James Keogh.
- Java EE and HTML5 Enterprise Application Development (Oracle Press) by John Brock, Arun Gupta, Geertjan Wielenga
- Struts: The Complete Reference, 2nd Edition by James Holmes

- ASP.NET Unleashed by Stephen Walther, Kevin Scott Hoffman, Nate Dudek
- Microsoft Visual C# 2013 Step by Step by John Sharp

GNU IMAGE MANIPULATION PROGRAMME

1. To familiarize the students with the underlying concepts of digital images.
2. To make the students know how to enhance images and prepare them for printing and publishing.

SYLLABUS **4 credits**

A. Theory

UNIT I Imaging Concepts and Graphic Formats: Pixel, Resolution, File Size, Image Compression, Raster & Vector Images, Color Model.

UNIT II Capturing and Creating Images: Saving Images, Scanning Images, Familiarization with GIMP Interface.

UNIT III Settings: Foreground and Background Colors, Grid Properties.

UNIT IV Image Manipulations: Resizing images, Cropping images, Moving and Copying images, Rotating and flipping images.

UNIT V Working with Text: Creating and editing text, Formatting Text, Applying text wraps.

UNIT VI Tools: Drawing tools, Painting tools.

REFERENCE BOOKS

- Kay Richter, GIMP 2.8- Buch (e-book)
- Olivier Lecarme and Karine Delvare, The Book of GIMP, A complete Guide to Nearly Everything, Kindle Edition

B. Practicum

Students are required to implement a project based on learned concepts.

7. Curriculum Alignment Matrix

Curriculum Alignment Matrix lists the learning objectives against the courses in the program and it becomes clear where assessment of student learning should occur. The curriculum alignment matrix becomes the basis of the assessment plan. With this, faculty and/or assessment coordinator can determine what student artifact or work sample (signature assignment or other assignment(s)) can be used to measure progress towards the learning objectives and/or when the assessment will take place. In addition, the matrix will help point out any gaps in the curriculum. The exercise of building and reviewing a curriculum alignment matrix encourages reflection on the curriculum and can lead to better integration among courses.

Curriculum Alignment Matrix is a table with one row for each learning outcome and one column for each course or required event/experience. Faculty identify where key learning outcomes are *introduced* (I), *reinforced* with the opportunity to practice (R or P), and where *mastery* (M) is achieved at the senior or exit level. When the matrix is complete, the program can identify where assessment evidence (A) should be gathered. In addition to courses, faculty should include any other required events/experiences (e.g., internships, department symposium, national licensure exams).

TABLE VI: Curriculum Alignment Matrix for BSc with Computer Science

Course Type	Course Name	PLO-									
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
CC-1A	Programming Methodology	I	I	-	I	-	-	I	-	I	-
CC-2A	Data Structures	R	-	-	R	-	I	I	-	-	-
CC-3A	Operating System	R	I	-	-	I	-	I	I	-	-
CC-4A	Database Management System	-	R	-	M	-	I	R	I	I	I
DSE-1A Any One	Software Engineering	M	R	-		I	I	R	I	I	-
	Computer Ethics	-	-	-	-	I	I	-	-	I	-
	Computer Organization &Architecture	I	I	-	-	-	I	-	I	-	-
	Computer Networks	-	M	I	-	R	R	I	R	-	I
DSE-2A Any One	Data Mining	M	-	-	M	-	R	M	-	-	R
	Internet of Thing	-	I	-	M	R	M	R	R	-	R
	Artificial Intelligence	M	-	I	-	-	M	R	-		R
	Computer Graphics	M	M	-	-	-	I	-	-	-	R
SEC-3A Any one	MATLAB Programming	R	M	-	R	-	R	I			
	Programming in Java	R	M	-	R	-	R	I	-	-	-
	Python Programming	R	M	-	R	-	R	I			
SEC-4A Any one	Web Programming	-	R	I	-	-	M	R	-	R	I
	Mobile Application Development	-	I	I	M	R	M	M	-	-	I
	Cloud Computing	-	R	I	M	R	M	M	-	-	I

TABLE VII: Curriculum Alignment Matrix for BSc (Hons) in Computer Science

	Course Name	**PLO-**														
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**
CC	Prog Methodology	I	I	-	I	-	-	I	-	I	-	I	I	-	-	-
	Comp System Arch	I	I	I	-	I	-	I	I	I	-	-	-	-	-	-
	Data Structures	R	-	-	R	-	I	I	-	-	-	I	-	-	-	-
	Discrete Structure											-	-	-	-	-
	Operating System	R	I	-	-	I	-	I	I	-	-	I	I	-	-	-
	Algorithm	M	M	I	M	-	R	-	-	-	-	R	I	-	-	-
	Computer Graphics	M	M	-	-	-	I	-	-	-	R	-	I	-	-	-
	Computer Networks	-	M	I	-	R	R	I	R	-	I	R	I	-	-	-
	Software Engineering	-	R	-	M	-	I	R	I	I	I	I	I	-	-	-
	DBMS	-	R	-	M	-	I	R	I	I	I	R	I	-	-	-
	Object-Oriented Prog	R	M	-	R	-	R	I	-	-	-	I	I	-	M	M
	Internet Technologies	M	-	I	-	-	M	R	-	-	R	R	R	I	M	M
	Artificial Intelligence	M	-	I	-	-	M	R	-	-	R	M	M	R	M	M
	Computer Graphics	M	-	I	-	-	M	R	-	R	R	M	R	-		
	Machine Learning	M	-	M	-	-	M	M	-	-	R	M	M	M	M	M
SEC	MATLAB Programming	R	M	-	R	-	R	I		I		-	I	I	I	I
	Programming in Java	R	M	-	R	-	R	I	-	-	-	I	I	I	I	I
	Python Programming	R	M	-	R	-	R	I		-		-	I	I	I	I
	Web Programming	-	R	I	-	-	M	R	-	R	I	-	I	I	I	I

	Mobile Application Development	-	I	I	M	R	M	M	-	-	I	-	I	I	I	I
DSE	Cloud Computing	-	R	I	M	R	M	M	-	-	I	R	M	R	R	R
	Image Processing	M	-	R	-	-	M	R	-	-	R	M	M	R	M	M
	Data Mining	M	-	R	-	-	M	R	-	-	R	M	R	R	M	M
	Data Analytics	M	-	R	-	-	M	R	-	-	R	M	R	R	R	M
	Internet of Things	M	-	R	-	-	M	R	-	-	R	R	R	R	M	M

8. Teaching-Learning Process

The teaching-learning process should be in-line with the course objective and outcomes. Teaching has to ensure that the suggested outcomes are ensured for each course and overall programme. Teaching-aids should be used wherever required to facilitate proper and impactful learning. Blended learning is recommended with the use of MOOC platforms and classroom teaching.

To meet the set objectives of the course and enable students achieve the expected outcomes of the course the teaching-learning process should be appropriately chosen. Though the teachers are best positioned to create innovative models suitable for teaching the course, certain well accepted and widely tested processes are suggested to achieve the desired outcomes

CLASSROOM TEACHING - Regular classroom and face to face teaching and tutorialscan be primarily used for imparting theoretical foundations of Computer Science. Applications of the same may be explained from time to time so that the student can appreciate the theory.

LABORATORY - Lab exercises in programming and usage of package / software tools should be made mandatory and integral part. Open source software/Packages should be preferred over proprietary tools wherever available.

SEMINARS - Guest lectures and seminars involving industry experts and eminent teachers should be arranged to help the students understand the practices in the industry and developments in the field.

MOOCS - Teacher should choose appropriate lecture materials and videos on similar courses available online through Massive Open Courses Online in the world wide web (such as NPTEL) to provide good perspective of the course and usecases and promote blended learning.

PROJECT - Wherever possible the laboratory assignments can be designed in the form of a mini project. For example, the database course lab assignments can be designed to build a complete system for library management. Similarly, summer/ Semester breaks can be utilized for guiding students to develop live projects with industry orientation/ industry problem. Teamwork work should be encouraged,

(1) ASSIGNMENTS - Home assignments should be designed to make student collect information from various sources and solve unfamiliar problems and make comparisons of solutions

(2) MAJOR PROJECT - The major project should be defined based on the student proposals keeping in mind that opportunity to demonstrate the knowledge and skills gained during the course. One-One mentoring support should be provided.

(3) Simulation - Packages to provide simulated environments to teach various components of networking and hardware working should be used wherever feasible.

9. Assessment Methods

The committee recommends that assessment should be viewed not only merely as a testing by the institution to evaluate the students' progress, but also as a valuable tool for a student to learn what is expected of him/her, where their level of knowledge and skill is lacking, and perhaps most importantly, what he/she could do to improve these levels with the valuable inputs of the lecturers. Assessment methods are the strategies, techniques, tools and instruments for collecting information to determine the extent to which students demonstrate desired learning outcomes. In the Bachelor's programmes leading to degrees such as BSc with Computer Science and BSc(Hons) in Computer Science, the assessment and evaluation methods focus on testing the conceptual understanding of the basic ideas of computer hardware and software, development of programming skills and experimental techniques, retention and ability to apply the knowledge acquired to real-life applications, and to solve new problems and communicate the results and findings effectively. Based on the Learning Objectives defined for each course as proposed in detail, assessment methods can be designed to monitor the progress in achieving the Learning Objectives during the course and test the level of achievement at the end of the course. Several methods can be used to assess student learning outcomes. Relying on only one method to provide information about the program will only reflect a part of students' achievement.

Modular Assessment

As the courses are broken up into a smaller more cohesive learning outcomes a module will consist of a number of these smaller, finer grained assessments of which the majority can be considered to be formative assessments that aid the learning process rather than assessments aimed at solely being used to evaluate the student.

Continuous Assessment

The continuous assessment occurs on a regular and continuous basis, it is an ongoing formative and summative process, involves the monitoring of students, is integrated with teaching, involves a systematic collection of marks or grades into a final score, may be used to determine the students' final grades.

Direct methods of assessment ask students to demonstrate their learning while indirect methods ask students to reflect on their learning. Tests, essays, presentations, etc. are generally direct

methods of assessment, and indirect methods include surveys and interviews. For each Learning Objective, a combination of direct and indirect assessment methods should be used.

Formative Assessment

While *formative assessment* is to gather feedback from formal or informal processes that can be used by the instructor and the students to gather evidence for the purpose of improving learning, *summative assessment* measures the level of success or proficiency that has been obtained at the end of an instructional unit, by comparing it against some standard or benchmark. Nevertheless, the outcome of a *summative assessment* can be used formatively when students or faculty use the results to guide their efforts and activities in subsequent courses. Daily programming assignments or home-assignments is a good way of implementing *formative assessment* and gives an idea of how well the students understood and could apply each programming concept. Another way of *formative assessment* can be that at the end of each class period, a student response system can be used to ask students one or more questions about the topic taught on that day. Regular tutorial Assignment, Term-paper, Seminar Presentation, Surprise Quizzes, Open-book Quizzes should be adopted for formative assessments. It is suggested that 25-30% weightage be given *Formative Assessments* in case of theory components while 30-40% weightage be given to the Programming/Laboratory/Projects/Dissertation components of the various courses.

During the semester, at least three smaller formative assessments shall be given for each course. To pass a course a student had to achieve marks between 70% in two of the assessment opportunities. The philosophy is that the student could fail one opportunity and take the experience gained from that opportunity to pass subsequent assessments.

Summative Assessment

For the traditional summative assessment, it is the semester tests based. The students need to attend two semester tests which consist of half of the content they learned for each test. Students are admitted to an examination for individual courses if they attain the minimum semester mark of 40%. Summative Assessment for the theory papers, can be a combination of Mid -Semester Test, Individual /Team Project report, Oral Presentations of Seminar/Projects, Viva -Voce Examination for dissertation and End Semester closed book examination. Summative Assessment methods shall be different for theory courses and Practical Courses.

It is suggested that the examination questions should be asked keeping the learning outcomes in mind and also covering all the Units. Term papers, problem solving assignments, Lab

projects, Internship experience, group projects are recommended for achieving the expected outcomes. Wherever possible, students need to do minor projects in practical classes to learn the technology and also to apply the technology for problem solving. As this is a technology oriented programme and new technologies are introduced quite often, care should be taken to familiarize the students with the recent advances through seminars or term papers and case studies. This should be given due weightage during continuous evaluation process. To achieve this objective, the following are suggested

(i) The end examination papers should be covering all units of the syllabus. Questions should be balanced and evaluate the comprehension, analytical and problem-solving skills.

(ii) The students should be evaluated on teamwork in addition to the technical skills through projects.

(iii) Ability to self-learning and solving new problems should be assessed through assignments, Seminars and project work.

(iv) It is recommended that 25-30% weightage of marks shall be devoted for formative assessment.

(v) It is recommended that 40% weightage be given for practical and laboratory work.

(vi) Peer evaluation component is recommended for project evaluation and seminar.

(vii) Online course certification should be encouraged and equivalent grade for the same need to be worked to achieve the outcome of self-learning.

10. Keywords

Learning Outcome, Graduate Aptitude, Qualification Descriptor, Generic Elective, Skill Enhancement, Core Compulsory Courses, Discipline Specific Elective, Summative Assessment, Formative Assessment, Curriculum Alignment Matrix.

****