SEMESTER I     Programming in C

<table>
<thead>
<tr>
<th>Sub Code: MCA11</th>
<th>CIE:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hours/ Week: 04</td>
<td>SEE:50</td>
</tr>
<tr>
<td>Total Hours: 50</td>
<td>Exam Hours: 03</td>
</tr>
</tbody>
</table>

COURSE OBJECTIVES:

- To initiate the students into the programming concepts and logical thinking
- Understanding the basic principles of C programming
- To understand the concept of variables, control statements and structures, functions, arrays, structures, pointers and files
- To be able to perform operation on files using pointers

MODULE 1
Introduction to C: Algorithms, Flow Charts, C structure, Variables, Data types, Constants, Declarations, Operators, Precedence, Associativity, Order of evaluation, Type conversion, Storage classes, Programming Examples 7 Hours

MODULE 2
Input and Output Statements: Input and output statements – scanf, getchar, gets, printf, putchar, puts; Control Statements – if, else-if, switch, Control Structures – while, for, do-while, break and continue, goto, Programming Examples 9 Hours

MODULE 3
Arrays: Arrays – Single dimension, Two dimensional, Multi dimensional Arrays, Strings, Programming Examples. 8 Hours

MODULE 4
Functions: Functions, Categories of functions, Pointers, Pointer arithmetic, call by value, Pointer Expression, Pointer as function arguments, recursion, Passing arrays to functions, passing Strings to functions, Call by reference, Functions returning pointers, Pointers to functions, Programming Examples. Structures and Unions – defining, declaring, initialization, accessing, comparing, operations on individual members; array of structures, structures within structures, structures and functions, pointers and structures, Programming Examples 13 Hours

MODULE 5
Files: Files – defining, opening, closing, input and output operations, error handling, random access; Command line arguments; Dynamic Memory Allocation – definition, malloc, calloc, realloc, free, dynamic arrays; Preprocessor – definition, macro substitution, file inclusion, compiler control directives, Programming Examples. Operation on Bits: Bit fields, operators, shifting operators, using bits as mask, application in assembly language. Introduction to Data Structures, Abstract Data Types, Queue, Stack, Linked list 13 Hours
COURSE OUTCOMES:

Upon successful completion of the course, the student will be able to

- Understand the basic concepts of C
- Apply programming skills starting with simple tasks and gradually develop file manipulation activities
- Choose appropriate data types such as arrays, structures, pointers and files and develop an interactive application
- Develop projects – system oriented and application oriented

Text Books
1. Let us C, Yashwant Kanetkar, BPB Publications
2. Programming with C, Balaguruswamy
3. The C Programming Language, Brian W Kernighan, Dennis M Ritchie, PHI, 2nd Edition

Reference Books
2. Simplifying C, Harshal Arolkar, Sonal Jain, Wiley Publications
3. Head First C, David Griffiths, & Dawn Griffiths, O’Riley.
SEMESTER I          Discrete Mathematical Structures

<table>
<thead>
<tr>
<th>Sub Code:</th>
<th>CIE:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCA12</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hours/ Week:</th>
<th>SEE:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total Hours:</th>
<th>Exam Hours:</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>03</td>
</tr>
</tbody>
</table>

Course objectives

- Recognizes logic by constructing direct and indirect proofs
- Infers logical reasoning to solve variety of problems
- Constructs induction proofs by summation, inequalities and divisibility arguments
- Identifies different set notations
- Comprehends cardinality, finiteness and determine the association between them
- Demonstrates different types of functions and their connection between cardinality
- Applies and analyses usage of graphs.

MODULE 1

**Fundamentals of Logic:** Basic Connectives and Truth Tables, Logic Equivalence : The laws of Logic, Logical Implications: Rules of Inference, The use of Quantifiers, Quantifier Definitions, Proofs of Theorems.

14 Hours

MODULE 2

**Set Theory:** Sets and Subsets, Set Operations and the Laws of Set Theory, Counting and Venn Diagrams, Principles of Inclusion and Exclusion, The rules of sum and product, Permutations and Combinations with repetition

9 Hours

MODULE 3


9 Hours

MODULE 4

**Relations and Functions:** Cartesian products and Relations, Functions-Plain and One-to-One, Onto Functions, The Pigeon-hole principle, Function composition and inverse functions, Properties of Relations, Computer recognition- Zero One Matrices and Directed graphs, Posets and Hasse Diagrams.

11 Hours
MODULE 5
Graph Theory and Trees: Basic concepts, Subgraphs, degrees of Vertices, paths and Connectedness, automorphism of a simple graph, Line graph, Operations on graphs. 9 Hours

Course outcome-
Upon successful completion of the course, the student will be able to
- Comprehend the proofs and definitions represented
- represent different set of entities using Venn diagrams
- Apply functions to relate one set of object to another
- Constructs graphs

Text Books

Reference books
SEMESTER I  Digital Logic and Computer Architecture

<table>
<thead>
<tr>
<th>Sub Code:</th>
<th>CIE:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCA13</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hours/ Week:</th>
<th>SEE:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total Hours:</th>
<th>Exam Hours:</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>03</td>
</tr>
</tbody>
</table>

COURSE OBJECTIVE:

- To familiarize with the basic concepts of digital systems and logic.
- Design sequential and combinational circuits
- Learn computer architecture and various designs.
- Make use of the basics of assembly language

MODULE 1
Digital Computer and Information: Digital Computers, binary numbers, decimal numbers, base of number system. Base Conversion, 1’s compliment and 2’s complement, addition and subtraction of numbers using complements, Arithmetic Operations, Decimal Codes, Alphanumeric Codes, Code conversion.
8 Hours

MODULE 2
12 Hours

MODULE 3
11 Hours

MODULE 4
Memory and Programmable Logic Devices: Memory and Programmable Logic Device, Random-access Memory, RAM integrated Circuits, Array of RAM ICs.
8 Hours

MODULE 5
Instruction Set Architecture: Computer Architecture Concepts, Operand Addressing, Addressing Modes, Instruction set Architectures, Data manipulation Instructions, Floating point Computation, Program Control Instruction, Program interrupts.

COURSE OUTCOMES:

Upon successful completion of the course, the student will be able to

- Evaluate number systems including computer arithmetic.
- Distinguish between working principles of electronic circuits e.g. flipflops, registers and counters.
- Distinguish between the basics of systems topics: single-cycle (MIPS), multi-cycle (MIPS), parallel, pipelined, superscalar, and RISC/CISC architectures, cost-performance issues and design trade-offs in designing, quantitative performance evaluation of computer systems.
- Explain the cache subsystem, assembly language programming, representation of data, addressing modes, instructions sets, basic knowledge the design of digital logic circuits and apply to computer organization.
- Visualize the working of file system, Secondary storage organization and thereby engage in problem solving.

Text books:

2. Morris Mano , Digital logic and computer design: PHI 23’ reprint October2000

Reference books:

SEMESTER I      Introduction to UNIX Programming

<table>
<thead>
<tr>
<th>Sub Code: MCA14</th>
<th>CIE:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hours/ Week: 04</td>
<td>SEE:50</td>
</tr>
<tr>
<td>Total Hours: 50</td>
<td>Exam Hours: 03</td>
</tr>
</tbody>
</table>

Course Objective:

The aim of this course is to enable the student to:

- To define and classify the approaches of an operating system.
- Illustrate the Unix Operating System in detail and to show how to work in Unix
- To understand the concept of basic commands, files, filters and process
- Application of the concepts in shell scripts
- Demonstrate the privileges of administrator
- Discuss and demonstrate the advanced filter

MODULE 1
UNIX - Introduction, History, Architecture, Experience the Unix environment, Basic commands ls, cat, cal, date, calendar, who, printf, tty, sty, uname, passwd, echo, tput, bc, script, spell and ispell, UNIX File System: The file, what’s in a filename? The parent-child relationship, pwd, the Home directory, absolute pathnames, using absolute pathnames for a command, cd, mkdir, rmdir, Relative pathnames, The UNIX file system.

12 Hours

MODULE 2
Introduction to the Shell: Introduction to Shell Scripting, Shell Scripts, working with Vi editor, read, Command Line Arguments, Exit Status of a Command, The Logical Operators && and ||, exit, if, and case conditions, expr, sleep and wait, while, until, for, $, @, redirection. The here document, set, trap, Sample Validation and Data Entry Scripts.

8 Hours

MODULE 3
Basic File Attributes: ls – l, the –d option, File Permissions, chmod, Security and File Permission, users and groups, security level, changing permission, user masks, changing ownership and group, FileAttributes, More file attributes: hard link, symbolic link, umask, find.
Simple Filters -Pr, head, tail, cut, paste, sort, uniq, tr commands, Filters using Regular Expression: grep & sed grep, Regular Expression, egrep, fgrep, sed instruction, Line Addressing, Inserting and Changing Text, Context addressing, writing selected lines to a file, the –f option, Substitution, Prosperities of Regular Expressions Context addressing, writing selected lines
to a file, the –f option, Substitution, Properties of Regular Expressions.

14 Hours

MODULE 4

7 Hours

MODULE 5
Awk-Advanced Filters:
Simple awk Filtering, Splitting a Line into Fields, printf, the Logical and Relational Operators, Number Processing, Variables, The –f option, BEGIN and END positional Parameters, get line, Built-in variables, Arrays, Functions, Interface with the Shell, Control Flow, Advanced Shell Programming, The sh command, export, cd, the Command, expr, Conditional Parameter Substitution, Merging Streams, Shell Functions, eval, Exec Statement.

9 Hours

Course Outcomes:
Upon successful completion of the course, the student will be able to

- Interpret the need of operating system
- Visualize the commands and develop interactive shell scripts
- Show the process execution and able to manage the system resources using super user commands
- Illustrate with case study and compare Unix operating system with other operating systems.

Text Book:
1. Your UNIX-The Ultimate Guide, Sumitabha Das, Tata McGraw Hill,

Reference Book:
1. “Unix Shell Programming”, Yashwant Kanetkar,
SEMESTER I

Introduction to Web Technologies

<table>
<thead>
<tr>
<th>Sub Code:</th>
<th>CIE:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCA15</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hours/ Week:</th>
<th>SEE:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total Hours:</th>
<th>Exam Hours:</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>03</td>
</tr>
</tbody>
</table>

Course Objectives

- Describe the design principles and techniques of web site design.
- Provide the students with the necessary knowledge and skills in using the various technologies and tools for developing web sites.
- Demonstrate the use of Python in developing applications using Web Application Framework (Django).

MODULE 1
Fundamentals-Web Foundations, Introduction to XHTML:

Cascading Style Sheets: Introduction, Levels of style sheets, Style specification formats, Selector forms, Property value forms, Font properties, List properties, Color, Alignment of text, The boxmodel, Background images, The <span> and <div> tags, Conflict resolution.

16 Hours

MODULE 2
The Basics of JavaScript:
Overview of JavaScript, Object orientation and JavaScript, Syntactic characteristics, Primitives, operations, and expressions, Screen output and keyboard input, Control statements, Object creation and modification, Arrays, Functions, Constructors, Pattern matching using regular expressions, Errors in scripts, Examples

6 Hours

MODULE 3
JavaScript and HTML Documents:
The JavaScript Execution Environment, The Document Object Model, Elements Access in Java Script, Events and Event Handling, Handling Events from Body Elements, Handling Events from Text Box and password Elements, The DOM2 Event Model, The navigator Object, DOM Tree Traversal and Modification

6 Hours
MODULE 4
Dynamic Documents with JavaScript:
Introduction, Positioning Elements, Moving Elements, Element Visibility, Changing Colors and Fonts, Dynamic Content, Stacking Elements, Locating the Mouse Cursor, Reacting to a Mouse Click, Slow Movement of Elements, Dragging and Dropping Elements.
6 Hours

MODULE 5
Introduction to Python:
16 Hours

Course Outcomes
At the end of the course student must:

- Demonstrate the knowledge and ability to apply the design principles, techniques and technologies to the development of creative websites.
- Apply JavaScript to add dynamic content to pages.
- Define well-structured, easily maintained, standards-compliant CSS code to present HTML pages in different ways.
- Develop web application framework (Django)

Text Books:

Reference Books:
SEMESTER I  C Programming Laboratory

<table>
<thead>
<tr>
<th>Sub Code:</th>
<th>CIE:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCA16</td>
<td></td>
</tr>
<tr>
<td>Hours/ Week: 03</td>
<td>SEE:50</td>
</tr>
<tr>
<td>Total Hours: 42</td>
<td>Exam Hours: 03</td>
</tr>
</tbody>
</table>

*The student must be capable of independently demonstrating his/her programming skill on the questions broadly based on the following concepts.*

**Course Objective**

- The student will learn the basics of the theory and practice of C programming
- The student will be introduced to the basics of programming and gradually be initiated to applying concepts of memory management and files
- The student will learn to apply logical skills and problem solving strategies

1. Implementation of the various Data Types with modifiers and type conversion in C

2. Write a program using nested if and switch... case structure thereby proving the difference between them *(Hints: Find the student grade or electricity tariff)*

3. Write a program using different Control structures in C *(Hints: use any combination of control structure statements including at least one control statements and one looping control statements. Print the gender of user. If boy, print ‘boy’ x times if age is less than or equal to z and y times if greater than z and similarly, print ‘girl’ n times if age is < z and m times if age is >=z)*

4. Simple program demonstrating the utilization of arrays *(Hints: declare an array, insert values in a list (1-D) and find the largest two numbers in array and find the average of all numbers.)*

5. Implementation of multidimensional arrays *(Hints: usage of multidimensional array perform the addition or subtraction, and multiplication)*

6. Implementation of functions:
   a) call by value; call by reference, passing of arrays
   b) Recursion

7. Demonstration of various user defined string operations (any one of string copy, string reverse, string compare, to upper case, to lower case, etc)

8. Exhibit the use of dynamic memory allocation using: malloc(), calloc(), free()
(Hints: find sum of n elements entered by user. To perform this program, allocate memory dynamically using malloc()/calloc() function.)

9. Demonstration of pointer operations-

(Hint: fetching content from memory 2 bytes ahead/behind pointer, position, Pointer arithmetic, pointer to a pointer, etc)

10. Implementation of structures and array of structures

(Hint: accept one student details and display and accept details of n students and display)

11. Implementation of Structures and Union.

(Bring the difference between them)

12. Demonstration of bitwise operations.

PART B

A simple application development approach to be performed independently by the student

Develop a simple menu driven program which will perform the following operation:

i. Create a file and display the contents
ii. Modify a file and display the original and modified contents
iii. Copy the file and display the contents
iv. Compare two files, displaying whether the two files are similar or not

Note: In the examination each student picks one question from the lot of all 12 questions in PART A and one sub section from PART B

The student will demonstrate his/her skill by developing a program on any one of the concepts in PART A and one of the options in PART B

Course Outcome
Upon successful completion of the course, the student will be able to

- Justify utilizing various concepts like arrays, functions, structures, pointers and files
- Support and build on self-learning skills choose/criticize a topic of interest on his/her own or as part of a team.
SEMESTER I

UNIX Programming Laboratory

<table>
<thead>
<tr>
<th>Sub Code:</th>
<th>CIE:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCA17</td>
<td></td>
</tr>
<tr>
<td>Hours/ Week: 03</td>
<td>SEE:50</td>
</tr>
<tr>
<td>Total Hours: 42</td>
<td>Exam Hours: 03</td>
</tr>
</tbody>
</table>

Course Objective:

- Familiarize the Unix environment
- Learn to work on Vi-editor
- Understand basic commands
- Apply commands and to write the shell scripts
- Understand the Filters and to use the basic filters
- Illustrate the file system concepts and to apply them in programs
- Learn the system management
- Distinguish between basic and advanced filters

Introduction to UNIX:

A. Explore the UNIX environment.

B. Explore vi editor with vim tutor. Perform the following operations using vi editor, but not limited to:
   1. Insert character, delete character, replace
   2. Save the file and continue working.
   3. Save the file a exit the editor
   4. Quit the editor
   5. Quit without saving the file
   6. Rename a file, copy a file
   7. Insert lines, delete lines,
   8. Set line numbers
   9. Search for a pattern
   10. Familiarise with arrow keys.

Exercise

1a. Write a shell script using expr command to read in a string and display a suitable message if it does not have at least 10 characters.

1b. Write a shell script that takes a valid directory name as an argument and recursively descend all the sub-directories, finds the maximum length of any file in that hierarchy and writes this maximum value to the standard output.

2a. Write a shell script that accepts a path name and creates all the components in that
Path name as directories. For example, if the script is named newpath, then the command newpath a/b/c/d should create directories a, a/b, a/b/c, a/b/c/d.

2b. Write a shell script that accepts two file names as arguments, checks if the Permissions for these files are identical and if the permissions are identical, output common permissions and otherwise output each file name followed by its permissions.

3a. Write a shell script which accepts valid log-in names as arguments and prints their corresponding home directories, if no arguments are specified, print a suitable error message.

3b. Create a script file3 called file-properties that reads a file name entered and outputs its properties.

4. Write shell script to implement terminal locking (similar to the lock command). It should prompt the user for a password. After accepting the password entered by the user, it must prompt again for the matching password as confirmation and if match occurs, it must lock the keyword until a matching password is entered again by the user, Note that the script must be written to disregard BREAK, control-D. No time limit need be implemented for the lock duration.

5a. Write a shell script that displays all the links to a file specified as the first argument to the script. The second argument, which is optional, can be used to specify in which the search is to begin. If this second argument is not present, the search is to begin in current working directory. In either case, the starting directory as well as all its subdirectories at 16 all levels must be searched. The script need not include any error checking.

5b. Write a shell script that accepts as filename as argument and display its creation time if file exist and if it does not send output error message.

6a. Write a shell script to display the calendar for current month with current date replaced by * or ** depending on whether the date has one digit or two digits.

6b. Write a shell script to find a file/s that matches a pattern given as command line argument in the home directory, display the contents of the file and copy the file into the directory~/mydir.

7a. Write a shell script that gets executed displays the message either “Good Morning” or
“Good Afternoon” or “Good Evening” depending upon time at which the user logs in.

7b. Write a shell script that accept a list of filenames as its argument, count and report occurrence of each word that is present in the first argument file on other argument files.

8a. Write a shell script that determine the period for which a specified user is working on system and display appropriate message.

8b Write a shell script that reports the logging in of a specified user within one minute after he/she log in. The script automatically terminate if specified user does not log in during a specified period of time.

9a. Write a shell script that accept the file name, starting and ending line number as an argument and display all the lines between the given line number.

9b. Write a shell script that folds long lines into 40 columns. Thus any line that exceeds 40 characters must be broken after 40th, a “\" is to be appended as the indication of folding and the processing is to be continued with the residue. The input is to be supplied through a text file created by the user.

10a. Write an awk script that accepts date argument in the form of dd-mm-yy and displays it in the form if month, day and year. The script should check the validity of the argument and in the case of error, display a suitable message.

10b. Write an awk script to delete duplicated line from a text file. The order of the original lines must remain unchanged.

11a. Write an awk script to find out total number of books sold in each discipline as well as total book sold using associate array down table as given below.

   Unix System - 60
   Web Technologies - 50
   Operating Systems - 30
   Computer Architecture - 44
   C Programming - 98
   Database Management Systems - 34

11b. Write an awk script to compute gross salary of an employee accordingly to rule given below.

   If basic salary is < 15000 then HRA=20% of basic & DA=45% of basic
If basic salary is $\geq 15000$ then HRA = 30% of basic & DA = 50% of basic.

Note: In the examination each student picks one question from a lot of all the 11 Questions

Course outcomes:

Upon successful completion of the course, the student will be able to

- Justify use of shell scripts with respect to concepts of commands, filters, process and file system
- Support system resources through super user commands
- Evaluate programs using advanced filters.
Course Objective:

- Describe the design principles and techniques of web site design.
- Provide the students with the necessary knowledge and skills in using the various technologies and tools for developing web sites.

1. Create an XHTML page to demonstrate the usage of
   a. Text Formatting tags,
   b. Links
   c. Images
   d. Tables
2. Develop and demonstrate the usage of inline and external style sheet using CSS
3. Develop and demonstrate a XHTML file that includes JavaScript script for the following problems:
   a) Input: A number n obtained using prompt
      Output: The first n Fibonacci numbers
   b) Input: A number n obtained using prompt
      Output: A table of numbers from 1 to n and their squares using `alert`
4. Develop and demonstrate using JavaScript, a XHTML document that displays random numbers (integers).
5. a) Develop and demonstrate, using JavaScript script, a XHTML document that collects the USN (the valid format is: A digit from 1 to 4 followed by two upper-case characters followed by two digits followed by two upper-case characters followed by three digits; no embedded spaces allowed) of the user. Event handler must be included for the form element that collects this information to validate the input. Messages in the alert windows must be produced when errors are detected.
   b) Modify the above program to get the current semester also (restricted to be a number from 1 to 6).
6. a) Develop and demonstrate, using JavaScript script, a XHTML document that contains three images, stacked on top of each other, with only enough of each showing so that the mouse cursor can be placed over some part of them. When the cursor is placed over the exposed part of any paragraph, it should rise to the top to become completely visible.
   b) Modify the above document so that when an image is moved from the top stacking position, it returns to its original position rather than to the bottom.
7. Develop using JavaScript script, an XHTML document that use of onload and onfocus events.
8. Write a Python program that reads a word, and prints the number of letters, vowels and the percentage of vowels in the word.
9. Write a python function to check a sentence to see if it is a pangram or not.
10. Write a python even driven program for file operations
11. Write a python program to check whether the given string is palindrome or not.
12. Write a Django program to read the contents & display it in the proper format.

Note: In the examination each student picks one question from the lot of all 12 questions.

Course Outcome:
Upon successful completion of the course, the student will be able to
- Choose and implement an appropriate planning strategy for developing websites.
- Support and produce functional, flexible, & versatile websites.
- Plan, locate, evaluate, & critically assess current & emerging technologies for developing websites.
- Apply good working knowledge of XHTML, CSS, and Python.
- Develop web application framework (Django)
Course Objective:
The aim of this course is to enable the student to:
- Understand the representation of data inside a computer in terms of memory and memory management.
- Understand the concepts like stacks, queues, lists, etc
- Achieve knowledge on different data structures and their applications.
- Understand the concepts of arrays linked lists, stacks, queues and trees
- Appreciate the importance of Data structures in operating systems and compilers
- Utilize different sorting techniques and appreciate their time and space complexities.

MODULE 1
Introduction to Data Structures:
Information and its meaning: Abstract Data Types, Sequences as Value Definitions, ADT for Data Types, Pointers and review of Pointers, Data Structures.
Arrays: the Array as an ADT, Using One-dimensional Arrays, Implementing One-Dimensional Arrays, Arrays as Parameters,
10 Hours

MODULE 2
Stacks and Queues:
Implementation of Stack, Array-based Implementation, Pointer-based Implementation, Applications of Stacks, Evaluation of Expressions, Evaluating Postfix Expression, Simulating Recursive Function using Stack, Passing Arguments, Return from a Function, Program to evaluate a postfix expression, Program to convert an expression from infix to postfix, Proving Correctness of Parenthesis in an Expression, Queue Implementation, Comparison with Array-based Implementation, Applications of Queues, the queue as ADT, Priority Queue, Circular Queues, Dequeue.
10 Hours

MODULE 3
Recursion:
Recursive definition and processes, Factorial function, Multiplication of natural numbers, Fibonacci sequence, Binary search, Properties of recursive definition or algorithm, Binary search, Towers of Hanoi problem.
4 Hours
MODULE 4
Linked List:

Linked lists, Inserting and removing nodes from a list, Linked implementations of stacks, get node and Free node operations, Linked implementation of queues, Linked list as a data Structure, Example of list operations, Header nodes, Array implementation of lists, Limitations of array implementation, allocating and freeing dynamic variables, Application of Linked list, Doubly linked list and implementation, Circular Linked list and implementation. 10 Hours

MODULE 5
Sorting and Searching Techniques:

Bubble sort, Quick sort, Selection sort, Tree Sorting: Binary Tree Sort, Heap Sort
Insertion Sorts: Simple Insertion, Shell Sort, Address Calculation Sort, Merge and Radix Sort.
Basic Search Techniques: Algorithmic Notations, Sequential searching, Indexed sequential search, Binary search, Interpolation search, Tree searching: Inserting into a Binary Search Tree, Deleting form a binary search tree Tree traversals, Binary Search Tree and Operations, Red-BlackTree, Threaded binary trees and operations. 16 Hours

Course outcomes:
At the end of the course, the student is

- Able to appreciate and utilize the concepts of data structure, data type and array data structure.
- Able to analyze algorithms and determine their time complexity.
- Able to implement linked list data structure to solve problems related to its application
- Able to understand and apply various data structure such as stacks, queues, trees and graphs to solve various computing problems using C-programming language.
- Able to implement and know when to apply standard algorithms for searching and sorting.

Text Books:
2. Data structures using D By Padma Reddy.

Reference Books:
SEMESTER II

Object Oriented Programming Using C++

<table>
<thead>
<tr>
<th>Sub Code: MCA22</th>
<th>CIE: 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hours/Week: 04</td>
<td>SEE: 50</td>
</tr>
<tr>
<td>Total Hours: 50</td>
<td>Exam Hours: 03</td>
</tr>
</tbody>
</table>

Course Objective

- To introduce object oriented programming concepts to automate the real time systems.
- To demonstrate polymorphism of different operators which can be used based on the context of the program.
- To overcome the drawbacks of structured programming by introducing inheritance concept which reduce the code complexity and increases the code reusability.
- To achieve code optimization using Generic functions and classes.
- To demonstrate compile time and run time exceptions, to handle abnormal program termination. These exceptions can be handled by the programmer using exception handling mechanisms.

MODULE 1

Introduction to OOPs, Modular Programming with Functions


MODULE 2

Classes & Objects and Operator Overloading

Introduction, Class Specification, Class Objects, access members, defining member functions, data hiding, constructors, destructors, parameterized constructors, static data members, functions, scope resolution operator, passing objects as arguments, returning objects, friend functions & classes, arrays of objects, Dynamic objects – Pointers to objects, Class members. Operator Overloading: Creating a Member Operator function, Binary operator overloading, concatenation of strings, strings comparison using operator overloading, overloading the assignment operator, overloading operators such as [ ], ->, increment & decrement operators, operator overloading using friend functions +, -, overloading input stream and output stream.
operators using friend function.

**MODULE 3**

**Inheritance Virtual functions & Polymorphism**

Base Class, Inheritance & protected members, protected base class inheritance, inheriting multiple base classes, Constructors, Destructors & Inheritance. Passing parameters to base Class Constructors, Granting access, Virtual base classes, Virtual function -Calling a Virtual function through a base class reference, Virtual attribute is inherited, Virtual functions are hierarchical, pure virtual functions, abstract classes, using Virtual functions, Early & late binding.

**MODULE 4**

**Templates, Exception Handling and I/O Streams**

Generic classes, a class template with more than one generic type, The power of templates. Exception Handling: Exception handling model, Exception handling constructs, list of exceptions, catch all exceptions, Handling uncaught exceptions. I/O Streams: IO Stream basics, output operator <<, input >>, additional I/O operators, overloading the output operator <<, overloading the input operator >>, file input & output, manipulators.

**MODULE 5**

**Standard Template Library:**

STL: An overview, containers, vectors, lists, maps.

**Course Outcome**

At the end of this course student is-

- Able to understand functions and parameter passing.
- Able to do numeric and string based computations.
- Able to understand dynamic memory allocation and pointers.
- Able to identify real time system's entities with OOPS concept.
- Able to create user defined data types and creating the corresponding objects.
- Able to implement the features of OOPS concept

**Text Books:**


**Reference Book:**
SEMESTER II

Operating Systems

<table>
<thead>
<tr>
<th>Sub Code:</th>
<th>CIE:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCA23</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hours/ Week:</th>
<th>SEE:50</th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total Hours:</th>
<th>Exam Hours:</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>03</td>
</tr>
</tbody>
</table>

Course Objective

The subject course provides

- To comprehend and understand the underlying principles, techniques and approaches of knowledge in operating systems.
- Illustrate inherent functionality and processing of program execution
- Identify the various underlying elements of operating system along with their interaction and provide services for execution of application software.

MODULE 1

Operating System Introduction:


08 Hours

MODULE 2

Process and CPU Scheduling


12 Hours

MODULE 3

Memory Management and Virtual Memory

Logical & Physical Address Space, Swapping, Contiguous Allocation, Paging, Structure of Page Table, Segmentation, Segmentation with Paging, Virtual Memory, Demand Paging,
Performance of Demanding Paging, Page Replacement Page Replacement Algorithms, Allocation of Frames, Thrashing, Case Studies: Linux, Windows. 10 Hours

MODULE 4
File System Interface


Mass Storage Structure – Overview of Mass Storage Structure, Disk Structure, Disk Attachment, Disk Scheduling, Disk Management, Swap space Management 10 Hours

MODULE 5
Deadlocks

System Model, Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection and Recovery from Deadlock.


Course Outcomes:

- Master functions, structures and history of operating systems
- Master understanding of design issues associated with operating systems
- Master various process management concepts including threads, scheduling, synchronization, deadlocks
- Master concepts of memory management including virtual memory
- Master system resources sharing among the users
- Master issues related to file system interface and implementation, disk management
- Be familiar with protection and security mechanisms
- Be familiar with various types of operating systems including Windows and Linux

TEXT BOOKS:


**REFERENCE BOOKS:**

5. An Introduction to Operating Systems, P.C.P. Bhatt, PHI.
Course Objectives –

To provide

- Introduce system software and differentiate between system software and application software. Emphasize on architecture of Simplified Instructional Computer and its extended version.
- Discuss the design and implementation of assemblers. There are certain fundamental functions that any assembler must perform, such as translation mnemonic operation codes to their machine language equivalents and assigning machine addresses to symbolic labels used by the programmer. This unit also teaches how to design of an assembler for Simplified Instructional Computer.
- Understand the various types of loaders and linkers available. Features and functions of loaders. Design and implementation of loaders and linkers. As an implementation example MS DOS Linker, Sun OS Linker and Cray MPP Linker are explained.
- Understand what is meant by macro, where and why it is used. Features of general-purpose macro processor. Design and implementation of macro processor. As and implementation of macro processor. As an implementation example Macro Assembler, ANSI C Macro and ELENA Macro Processor are explained.
- Introduce the concepts and principles of compiler design. Providing students with basic understanding of grammars and language definition. Introducing students to the various phases of designing a compiler. Introducing students to the various programming techniques and structures used in compiler construction.

MODULE 1
Machine Architecture


7 Hours

MODULE 2
Assemblers

Basic Assembler Function - A Simple SIC Assembler, Assembler Algorithm and Data Structures, Machine Dependent Assembler Features - Instruction Formats & Addressing Modes,

MODULE 3
Loaders and Linkers


MODULE 4
Macro Processor

Basic Macro Processor Functions - Macro Definitions and Expansion, Macro Processor Algorithm and Data Structures, Machine-Independent Macro Processor Features – Concatenation of Macro Parameters, Generation of Unique Labels, Conditional Macro Expansion, Keyword Macro ParametErs, Macro Processor Design Options – Recursive Macro Expansion, General- Purpose Macro Processors, Macro Processing Within Language Translators, Implementation Examples - MASM Macro Processor, ANSI C Macro Processor., ELENA macro processor. 9 Hours

MODULE 5
Compilers


Course outcome-

On Completion of this course, the students are able to

- Complete knowledge on what is System Software, able to distinguish the difference between system software and application software. Complete knowledge of SIC and SIC/XE
- Complete knowledge of the fundamental functions that any assembler must perform, such as translation mnemonic operation codes to their machine language equivalents and assigning machine addresses to symbolic labels used by the programmer. How to
design basic assembler for Simplified Instructional Computer, machine dependent 
assembler and machine independent assembler.

- Complete knowledge of various types of loaders and linkers. Features and functions of loaders. Design and implementation of loaders and linkers. Understand various Linkers and Loaders such as MS DOS Linker, Sun OS Linker and Cray MPP Linker are explained.

- Complete understand of what is meant by macro, where and why it is used. Complete knowledge and features of general-purpose macro processor. Design and implementation of macro processor.

- Complete understand the concepts and principles of compiler design. Understand context free grammars and language definition. Students will understand various phases and designing of a compiler.

Text Books:

Reference Books:

SEMESTER II
Database Management Systems
COURSE OBJECTIVES:

- Demonstrate the fundamentals of data models and conceptualize and depict a database system
- Make use of ER diagram in developing ER Model
- To Summarize the SQL and relational database design.
- Illustrate transaction processing, concurrency control techniques and recovery
- Inference the database design in the real world entities.

MODULE 1
Introduction

An example; Characteristics of Database approach; Actors on the screen; Workers behind the scene; Advantages of using DBMS approach; A brief history of Database Applications; When not to use a DBMS. Data Models, Schemas and Instances; Three-Schema Architecture and Data Independence; Database Languages and Interfaces; The Database System Environment; Centralized and Client-Server Architectures for DBMSs; Classification of Database Management Systems.

Entity-Relationship Model: Using High-Level Conceptual Data Models for Database Design; An Example Database Application; Entity Types, Entity Sets, Attributes and Keys; Relationship Types, Relationship Sets, Roles and Structural Constraints; Weak Entity Types; Refining the ER Design for COMPANY Database; ER Diagrams, Naming Conventions and Design Issues; Relationship Types of Degree Higher than Two, Relational Database Design Using ER-to-Relational Mapping.

14 Hours

MODULE 2
Relational Model and Relational Algebra

Relational Model Concepts; Relational Model Constraints and Relational Database Schemas; Update Operations, Transactions and Dealing with Constraint Violations; Unary Relational Operations: SELECT and PROJECT; Relational Algebra Operations from Set Theory; Binary Relational Operations: JOIN and DIVISION; Additional Relational Operations; Examples of Queries in Relational Algebra;

10 Hours

MODULE 3
SQL
SQL Data Definition and Data Types; Specifying Constraints in SQL; Schema Change Statements in SQL; Basic Queries in SQL; More Complex SQL Queries, Insert, Delete and Update Statements in SQL; Specifying Constraints as Assertions and Triggers; Views (Virtual Tables) in SQL; Additional Features of SQL; Database Programming: Issues and Techniques; Embedded SQL, Dynamic SQL; Database Stored Procedures and SQL / PSM.

Hours

MODULE 4
Database Design

Informal Design Guidelines for Relation Schemas; Functional Dependencies; Normal Forms Based on Primary Keys; General Definitions of Second and Third Normal Forms; Boyce-Codd Normal Form

Hours

MODULE 5
Transaction Processing


Hours

COURSE OUTCOMES:

On completion of this course, student is able to-

- Understand the basic concepts of the database and data models.
- design a database using ER diagrams and map ER into Relations and normalize the relations
- Acquire the knowledge of query evaluation to monitor the performance of the DBMS.
- Develop a simple database applications using normalization.
- Acquire the knowledge about different special purpose databases and to critique how they differ from traditional database systems.

Text Books:
2. Raghu Ramakrishnan, Johannes Gehrke, ’ Database management systems” McGraw Hill

REFERENCES:
Course Objective
- To understand the utilization of memory in an efficient and effective manner
- Develop skills to design and analyze data structures such as Lists, Stacks, Queues, Trees
- Build capability to identify and apply the suitable data structure for a given real world problem
- Appreciate the practical applications of data structures

Exercise
1. Program to perform stack implementation.
2. Program to convert from infix notation to Postfix to Prefix notations.
3. Simulate the working of Priority queue providing the following operations – Insert, Delete and Display.
4. Demonstrate recursion
   a. Solve Towers of Hanoi Problem
   b. Calculate the sum for a given number ‘n’ from 1 to n.
5. Implement linked lists and some operations on linked lists.
6. Implement Circular linked lists.
7. Implement
   a. Selection sort.
   b. Heap sort.
8. Create a binary tree and implement the tree traversal techniques of inorder, preorder and Postorder
9. Implement the search techniques of
   a. Linear Search
   b. Binary Search
10. Program Insertion, Deletion and Traversal In Binary Search Tree.

Course Outcome
At the end of this lab session, the student will
- Be fully aware of the concept of time and space efficiency of the data structure
- Be capable to analyze a problem and identify the appropriate data structure to be utilized for efficient performance
- Will have the required knowledge to appreciate memory management concepts in Operating Systems and also further pursue Analysis and design of algorithms, Compiler Design
Course Objectives:

- Demonstrate database handling process.
- Illustrate queries on the databases.

PART A:

Unit I SQL Practical

Data Definition Language: Create, Alter, Drop, Rename, Truncate
Data Manipulation Language: Insert, Update, Delete, Select 4 Hours

Unit II Data Control Language: Grant, Revoke, Roles

Transaction Control: Commit, Rollback, Savepoint
SQL SELECT Statements: Selecting All Columns, Selecting Specific Columns, Column Alias, Concatenation Operator, Arithmetic Operators, Comparison Conditions, Logical Conditions, ORDER BY Clause 4 Hours

Unit III Functions:

Single Row Functions, Character Functions, Number Functions, Date Functions, Conversion Functions, General Functions, Multiple Row Functions, Group Function
Subquery: Subquery, Types of Subquery, Group Function, Having Clause 4 Hours

Unit IV Joins:

Equijoins, Non-Equijoins, Joining Three Tables, Self Joins, Left Outer Joins, Right Outer Joins, Full Outer Joins, Cross Joins, Natural Joins
Other Concepts: Sequence, View, Index, Synonyms 4 Hours

Unit V Constraints:

Not Null, Unique Key, Primary Key, Foreign Key, Check, Dropping a Constraint, Enabling & Disabling 4 Hours

Course Outcomes:
On successful completion of this course, student will be:

- Able to make use of all SQL Statements to create and handle databases
- Able to plan to write and execute queries on the databases
- Design database based on the case study given.

PART B:

Mini Project:

Course Objectives:

- To teach database handling using real world entities (case study)
- To teach application DBMS concepts to apply in particular case studies.

Database project:

Use of Real World Application with Technological Application by using Open Source software application and Tool

Software / Tools: Mysql, Postgre SQL 9.0

Course Outcomes:
At the end of this course -
- Students should be able to design and handle database application
- Students should be able to maintain the transaction and maintenance of the DBMS Application.
- Students should be able to write and execute queries on the databases
Course Objective:

- To practice the fundamental programming methodologies in C++ programming language.
- To explore object-oriented programming features such as abstraction, polymorphism, inheritance, etc., using C++.
- To write reusable modules (collections of functions).

Exercise

1. Define a STUDENT class with USN, Name, and Marks in 3 tests of a subject. Declare an array of 10 STUDENT objects. Using appropriate functions, find the average of the two better marks for each student. Print the USN, Name and the average marks of all the students.

2. Write a C++ program to create a class called COMPLEX and implement the following overloading functions ADD that return a complex number:
   (i) ADD (a, s2) – where ‘a’ is an integer (real part) and s2 is a complex number
   (ii) ADD (s1, s2) – where s1 and s2 are complex numbers

3. Write a C++ program for scalar multiplication of two vectors using operator overloading.

4. Write a C++ program to create a template function for Bubble Sort and demonstrate sorting of integers and doubles.

5. Write a C++ program to create a class called LIST (linked list) with member functions to insert an element at the front and delete an element from the front of the list. Demonstrate all the functions after creating a LIST object.

6. Write a C++ program to create a class called STACK using an array of integers. Implement the following operations by overloading the operators ‘+’ and ‘--’:
   (i) s1 = s1 + element; where s1 is an object of the class STACK and element is an integer to be pushed on the top of the stack
   (ii) s1 = --s1 ; where s1 is an object of the class STACK. ‘--’ operator pops the element.
   Handle the STACK empty and full conditions. Also display the contents of the stack after each operation, by overloading the << operator.

7. Create a class called MATRIX using two-dimensional array of integers. Implement the following operations by overloading the operator == which checks the compatibility of two matrices to be added and subtracted. Perform the addition and subtraction by overloading the + and – operators respectively. Display the results by overloading the operator <<. If (m1==m2) then m3 = m1+m2 and m4 = m1-m2 else display error.

8. Write a C++ program to create a class called OCTAL which has the characteristics of an octal number. Implement the following operations by writing an appropriate constructor and an overloaded operator +.
   (i) OCTAL h = x; where x is an integer.
   (ii) int y = h + k; where h is an OCTAL object and k is an integer.
Display the OCTAL result by overloading the operator <<. Also display the values of h and y.

9. Write a C++ program to create a class template called QUEUE with member functions to add an element and to delete an element from the queue. Using the member functions, implement a queue of integers and double. Demonstrate the operations by displaying the contents of the queue after every operation.

10. Define a class SET with Data members: array of int, int variable to indicate number of elements in a SET object; and Member functions: to read element of a SET object, to print elements of a SET object, to find union of 2 objects of SET using operator overloading (S3=S1+S2), to find intersection of 2 objects of SET using operator overloading (S4= S1*S2). S1, S2, S3 and S4 are objects of SET. Use this class in a main function to show the above operations.

11. Write a C++ program to create a class called STUDENT with data members USN, Name and Age. Using inheritance, create the classes UGSTUDENT and PGSTUDENT having fields as Semester, Fees and Stipend. Enter the data for at least 5 students. Find the semester wise average age for all UG and PG students separately.

12. Write a C++ program to create a class called STRING and implement the following operations. Display the results after every operation by overloading the operator <<.
   (i) STRING s1 = “VTU”
   (ii) STRING s2 = “BELGAUM”
   (iii) STRING s3 = s1 + s2 (Use copy constructor)

13. Define a base class STACK1 which performs only push, pop, display operations. Override the above operations through a derived class STACK2 which takes care of STACK FULL & STACK EMPTY situations. Show how the objects of these classes use the above functions in a main function.

14. Create an abstract base class EMPLOYEE with data members: Name, EmpID and BasicSal and a pure virtual function Cal_Sal(). Create two derived classes MANAGER (with data members: DA and HRA) and SALESMAN (with data members: DA, HRA and TA). Write appropriate constructors and member functions to initialize the data, read and write the data and to calculate the net salary. The main() function should create array of base class pointers/references to invoke overridden functions and hence to implement run-time polymorphism.

15. Write a program to create a file to store some records and search for a particular record and display it.

Note: In the examination each student has to pick one question from a lot of all the 15 questions.
Course Outcome:

On successful completion of this course, student will be-

- Able to demonstrate functions and function overloading.
- Able to Creating class and objects in C++.
- Implementing inheritance, polymorphism and object relationship in C++.
- Able to make use of Constructors and destructors in programs.
- Able to demonstrate Friend functions and friend classes.
- Able to overload operators for the manipulation of user defined data types.