Raja N. L. Khan Women's College (Autonomous)

Syllabus and Scheme of Examination

for

B. C. A. (Honours)

Under

NEP-2020

with effect from 2023-24 session

Raja Narendra Lal Khan Women's College (Autonomous)

April 2024

Course Structure of Bachelor of Computer Applications (Honours)

Semester	Course Type	Credit	L-T-P	Marks
I	I. Major Paper: (1 Paper) 1.*Programming Fundamentals using C*	4 Credits	3-0-2	(Theory Exam + Practical Exam + Continuous Assessment + Attendance) = (40 + 20 + 10 + 5) = 75 Marks
	II. Minor Paper (1 Paper) 1. Mathematics	4 Credits	4-0-0	(Theory Exam + Continuous Assessment + Attendance) = (60 + 10 + 5) 75 Marks
	III. Interdisciplinary/Multidisciplinary Paper (1 Paper) ENVS	3 Credits	1-0-2	(Theory Exam + Report + Continuous Assessment + Attendance) = (30 + 10 + 5+5) = 50 Marks
	IV. Skill Enhancement Course (SEC) Paper: (1 Paper) 1. *Digital Electronics and IC Fabrication*	3 Credits	2-0-2	(Theory Exam + Practical Exam + Continuous Assessment) = (20 + 20 + 10) = 50 Marks
	V. Value Added Course (2 Papers) 1. Mental Health and Wellbeing, 2. Physical Fitness and Community Service	2 × 2 Credits = 4 Credits	2-0-0	(Theory Exam + Continuous Assessment) = (30 + 10 + 5+5) = 50 Marks
	VI. Ability Enhancement Elective (Skill Based Theory and Lab) 1. *English*	2 Credits	2-0-0	(Theory Exam + Continuous Assessment) = (40 + 10) = 50 Marks
	Total	Total Credit: 20		Total Marks:400
II	I. Major Paper: (1 Paper) 2. *Data Structure*	4 Credits	3-0-2	(Theory Exam + Practical Exam + Continuous Assessment + Attendance) = (40 + 20 + 10 + 5) = 75 Marks
	II. Minor Paper (1 Paper) 1. Mathematics	4 Credits	4-0-0	(Exam + Continuous Assessment + Attendance) = (60 + 10 + 5) 75 Marks

	III. Interdisciplinary/Multidisciplinary Paper (1 Paper) 2. *Climate Change & Disaster Management*	3 Credits	1-0-2	(Theory Exam + Report + Continuous Assessment + Attendance) = (30 + 10 + 5+5) = 50 Marks
	IV. Skill Enhancement Course (SEC) Paper: (1 Paper) 2. *Programming in Python*	3 Credits	2-0-2	(Theory Exam + Practical Exam + Continuous Assessment) = (20 + 20 + 10) = 50 Marks
	V. Value Added Course (2 Papers) 3. Indian Constitution 4. Indian Knowledge System	2 × 2 Credits =4 Credits	2-0-0	(Theory Exam + Continuous Assessment) = (30 + 10 + 5+5) = 50 Marks
	VI. Ability Enhancement Elective (Skill Based Theory and Lab) 1. *MIL*	2 Credits	2-0-0	(Theory Exam + Continuous Assessment) = (40 + 10) = 50 Marks
	VII. Community Engagement 1. *Community Engagement & Social Responsibility*	2 Credits (4 credit for those who want to exit after 2nd sem)	0-0-2	(Continuous Assessment) = 50 Marks
	Total	Total Credit: 22		Total Marks:450
III	I. Major Papers: (2 Papers) 3. *Object-Oriented Programming* 4. *Computer Architecture & Organization*	2 × 4 Credits =8 Credits	3-0-2	(Theory Exam + Practical Exam + Continuous Assessment + Attendance) = (40 + 20 + 10 + 5) = 75 Marks
	II. Minor Paper (1 Paper) 2. Physics / Economics	4 Credits	3-0-1 / 4-0-0	(Exam + Continuous Assessment + Attendance) = (60 + 10 + 5) 75 Marks
	III. Interdisciplinary/Multidisciplinary Paper (1 Paper) 3. *Introduction to Computer & Artificial Intelligence*	3 Credits	1-0-2	(Theory Exam + Report + Continuous Assessment + Attendance) = (30 + 10 + 5+5) = 50 Marks

	IV. Skill Enhancement Course (SEC) Paper: (1 Paper) 3. *Web Technology*	3 Credits	2-0-2	(Theory Exam + Practical Exam + Continuous Assessment) = (20 + 20 + 10) = 50 Marks
	V. Ability Enhancement Elective (Skill Based Theory and Lab) 2. *English*	2 Credits	2-0-0	(Theory Exam + Continuous Assessment) = (40 + 10) = 50 Marks
	Total	Total Credit: 20		Total Marks:375
IV	I. Major Papers: (3 Papers) 5. *Operating Systems* 6. *Design & Analysis of Algorithms* 7. *Software Engineering*	3 × 4 Credits =12 Credits	3-0-2	(Theory Exam + Practical Exam + Continuous Assessment + Attendance) = (40 + 20 + 10 + 5) = 75 Marks
	II. Minor Paper (1 Paper) . 2. Physics/Economics	4 Credits	3-0-1 / 4-0-0	(Exam + Continuous Assessment + Attendance) = (60 + 10 + 5) 75 Marks
	V. Ability Enhancement Elective (Skill Based Theory and Lab) 2. *MIL*	2 Credits	2-0-0	(Theory Exam + Continuous Assessment) = (40 + 10) = 50 Marks
	VI. Community Engagement 2. *Community Engagement & Social Responsibility*	2 Credits	0-0-2	(Continuous Assessment) = 50 Marks
	Total	Total Credit: 20		Total Marks:400
V	I. Major Papers: (4 Papers) 8. *Database Management Systems* 9. *Computer Networks* 10. *Theory of Computation* 11. *Artificial Intelligence*	4 × 4 Credits =16 Credits	3-0-2	(Theory Exam + Practical Exam + Continuous Assessment + Attendance) = (40 + 20 + 10 + 5) = 75 Marks
	II. Minor Paper (1 Paper) 3. Mathematics	4 Credits	3-0-1 / 4-0-0	(Exam + Continuous Assessment + Attendance)

				= (60 + 10 + 5) 75 Marks
	Total	Total Credit: 20		Total Marks:375
VI	I. Major Papers: (4 Papers) 12. *Computer Graphics* 13. *Machine Learning* 14. *Compiler Design* 15. *Cyber Security*	4 × 4 Credits =16 Credits	3-0-2	(Theory Exam + Practical Exam + Continuous Assessment + Attendance) = (40 + 20 + 10 + 5) = 75 Marks
	II. Minor Paper (1 Paper) 3. Physics/Economics	4 Credits	3-0-1 / 4-0-0	(Exam + Continuous Assessment + Attendance) = (60 + 10 + 5) 75 Marks
	III. Community Engagement 1. *Summer Internship*	2 Credits	0-0-2	(End Semester Assessment) = 50 Marks
	Total	Total Credit: 20		Total Marks:425
VII	I. Major Papers: (4 Papers) 16. to be finalized 17. to be finalized 18. to be finalized 19. to be finalized II. Minor Paper (1 Paper) 4. Mathematics	Total Credit: 20		
VIII	I. Major Papers: (2 Papers) 20. to be finalized 21. to be finalized II. Minor Paper (1 Paper) 4. Physics/Economics III. Research Project/Dissertation* (*additional 3 core paper of 4 credit each to be studied for students without Honours with Research Degree	Total Credit: 24		

Programme Objectives (PO)

PO 1: To provide students with a solid foundation in Computer Application fundamentals necessary to analyze the requirements of the software, design and create innovative software products and solutions for real life problems.

PO 2: To provide exposure to emerging technologies to work as teams on multidisciplinary

PO 3: To prepare the students for a successful professional career as developer, scientist, teacher, administrator or an entrepreneur and work with values & social concern bridging the digital divide and meeting the requirements of Indian and multinational companies.

BCAHMJ101: Programming Fundamentals using C Theory: 40 Lectures

	Course Outcome
1.	Know the procedural programming paradigm
2.	Develop the skill to understand programming logic for solving a computing problem
3.	Understand syntax of C programming language
4.	Apply procedural programming skill to solve a given problem using C

Unit 1: **Introduction to Programming** **(3 Lectures)**

Introduction - Introduction to computer system, Types of software, Operating system as user interface, algorithm, flowchart, High-level language, low-level language, Need of Compiler and interpreter, Different programming languages, Overview of Procedural Programming, Use of main() function, Compiling and Executing Simple Programs in C.

Unit 2: **Data Types, Variables, Constants, Operators and Basic I/O** **(4 Lectures)**

Declaring, Defining and Initializing Variables, Scope of Variables, Using Named Constants, Keywords, Data Types, Casting of Data Types, Operators (Arithmetic, Logical and Bitwise), Using Comments in programs, Character I/O (getc, getchar, putc, putchar etc), Formatted and Console I/O (printf(), scanf()), using basic header files (stdio.h, conio.h etc).

Unit 3: **Expressions, Conditional Statements and Iterative Statements** **(8 Lectures)**

Simple Expressions in C (including Unary Operator Expressions, Binary Operator Expressions), Understanding Operators, Precedence in Expressions, Conditional Statements (if construct, switch-case construct), Understanding syntax and utility of Iterative Statements (while, do-while, and for loops), Use of break and continue in Loops, Using Nested Statements (Conditional as well as Iterative)

Unit 4:**Functions and Arrays** **(9 Lectures)**

Utility of functions, Call by Value, Call by address, Functions returning value, Void functions, Return data type of functions, Functions parameters, Differentiating between Declaration and Definition of Functions, Recursion.

Creating and Using One Dimensional Arrays (Declaring and Defining an Array, Initializing an Array, Accessing individual elements in an Array, Manipulating array elements using loops), Use Various types of arrays (integer, float and character arrays / Strings) Two- dimensional Arrays (Declaring, Defining and Initializing Two Dimensional Array, Working with Rows and Columns), Introduction to Multi-dimensional arrays. Different storage classes and scope of variables.

Unit 5: **Derived Data Types (Structures and Unions)** **(4 Lectures)**

Understanding utility of structures and unions, Declaring, initializing and using simple structures and unions, Manipulating individual members of structures and unions, Array of Structures, Individual data members as structures, Passing and returning structures from functions, Structure with union as members, Union with structures as members, enum.

Unit 6: **Pointers** **(4 Lectures)**

Understanding a Pointer Variable, Simple use of Pointers (Declaring and Dereferencing Pointers to simple variables), Pointers to Pointers, Pointers to structures, Problems with Pointers, Passing pointers as function arguments, Returning a pointer from a function, using arrays as pointers, Passing arrays to functions. Pointer arithmetic. Command Line Arguments.

Unit 7: **Dynamic Memory Allocation** **(4 Lectures)**

Differentiating between static and dynamic memory allocation, use of malloc, calloc, realloc and free functions, storage of variables in static and dynamic memory allocation

Unit 8: **File I/O, Preprocessor Directives** **(4 Lectures)**

Opening and closing a file (use of fopen, fclose), Reading and writing Text Files, using fscanf(), fprintf(), read() and write() functions, Random access in files, Understanding the Preprocessor Directives (#include, #define, #error, #if, #else, #elif, #endif, #ifdef, #ifndef and #undef), Macros.

Recommended Books

1. Dey, Bandyopadhyay, *C Programming Essentials*, Pearson Education.
2. Balagurusamy, *Programming in ANSI C,* Tata McGraw Hill.
3. Kerninghan, Ritchie, *The C Programming Language*, Pearson.
4. Kanetkar, *Let Us C*, BPB Publications.

Programming Fundamentals using C Lab Practical: 30 Lectures

	Course Outcome
1.	Learn the C programming syntax
2.	Develop programming logic
3.	Ability to write and execute C programs for solving a given problem
4.	Comprehend the compilation error messages and debugging
5.	Develop skill to trace execution of a program to fix errors in program

1. Write a program (WAP) to print the sum and product of digits of an integer.

2. WAP to reverse a number.

3. WAP to compute the sum of the first n terms of the following series S = 1+1/2+1/3+1/4+……

4. WAP to compute the sum of the first n terms of the following series S =1-2+3-4+5…………….

5. Write a function that checks whether a given string is Palindrome or not. Use this function to find whether the string entered by user is Palindrome or not.

6. Write a function to find whether a given no. is prime or not. Use the same to generate the prime numbers less than 100.

*7.*WAP to compute the factors of a given number.

8. Write a macro that swaps two numbers. WAP to use it.

9. WAP to print a triangle of stars as follows (take number of lines from user):

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

10. WAP to perform following actions on an array entered by the user:

i) Print the even-valued elements
ii) Print the odd-valued elements
iii) Calculate and print the sum and average of the elements of array
iv) Print the maximum and minimum element of array
v) Remove the duplicates from the array
vi) Print the array in reverse order

The program should present a menu to the user and ask for one of the options. The menu should also include options to re-enter array and to quit the program.

11. WAP that prints a table indicating the number of occurrences of each alphabet in the text entered as command line arguments.

12. Write a program that swaps two numbers using pointers.

13. Write a program in which a function is passed address of two variables and then alter its contents.

14. Write a program which takes the radius of a circle as input from the user, passes it to another function that computes the area and the circumference of the circle and displays the value of area and circumference from the main() function.

15. Write a program to find sum of n elements entered by the user. To write this program, allocate memory dynamically using malloc() / calloc() functions or new operator.

16. Write a menu driven program to perform following operations on strings:

a) Show address of each character in string
b) Concatenate two strings without using strcat function.
c) Concatenate two strings using strcat function.
d) Compare two strings
e) Calculate length of the string (use pointers)
f) Convert all lowercase characters to uppercase
g) Convert all uppercase characters to lowercase
h) Calculate number of vowels
i) Reverse the string

17. Given two ordered arrays of integers, write a program to merge the two-arrays to get an ordered array.

18. WAP to display Fibonacci series (i)using recursion, (ii) using iteration

19. WAP to calculate Factorial of a number (i)using recursion, (ii) using iteration

20. WAP to calculate GCD of two numbers (i) with recursion (ii) without recursion.

21. Write a menu-driven program to perform following Matrix operations (2-D array implementation):
a) Sum b) Difference c) Product d) Transpose

22. Create a structure Student containing fields for Roll No., Name, Class, Year and Total Marks. Create 10 students and store them in a file.

23. Write a program to retrieve the student information from file created in previous question and print it in following format: *Roll No. Name Marks*

24. Copy the contents of one text file to another file, after removing all white spaces.

25. Write a function that reverses the elements of an array in place. The function must accept only one pointer value and return void.

26. Write a program that will read 10 integers from user and store them in an array. Implement array using pointers. The program will print the array elements in ascending and descending order.

BCAHSEC01: Digital Electronics and IC Fabrication Theory: 30 Lectures

	Course Outcome
1.	Comprehension of number systems
2.	Understand the integer and floating-point representation and binary arithmetic
3.	Minimization of Boolean functions
4.	Comprehend different basic combinational and sequential circuits
5.	Design different combinational and sequential logic circuits
6.	Understand the IC fabrication process

Unit 1: Number System- Different number systems like binary, octal, and hexa-decimal. Hamming distance between code words, Gray code, Excess-3 code. (3 Lectures)

Unit 2: Integer and Floating-Point Numbers: Signed integers in sign-magnitude, 2's complement and 1's complement representations. Floating-point number representations. (4 Lectures)

Unit 3: Binary arithmetic: Addition, Subtraction, and Multiplication. (4 Lectures)

Unit 4: Logic Gates: AND, OR, NOT, EX-OR, EX-NOR. Truth tables. Expressions. Universal logic gates. Associativity of NAND, NOR Applications of EX-OR. (4 Lectures)

Unit 5: Boolean algebra, Boolean functions, min terms and max terms, sum of products, product of sums, minimization of Boolean function using Karnaugh map. (4 Lectures)

Unit 6: Combinational circuits: Decoder, encoder, multiplexor, de-multiplexor. Representation of Boolean function using multiplexor. Half adder, full adder, half subtractor, 2-bit magnitude comparator. (4 Lectures)

Unit 7: Sequential circuits: S-R, D, J-K, T flip flops, Counters, Shift registers. (4 Lectures)

Unit 8: IC Fabrication: Logic families – TTL, ECL, CMOS. How ICs are built: HDL, synthesis, placement, routing, fabrication, packaging and testing. (3 Lectures)

Recommended Books

1. Rajaraman, Adabala, *Fundamentals of* Computers, PHI.
2. Sinha, Sinha, *Computer Fundamentals: Concepts, Systems & Applications*, BPB.
3. Mano, *Digital Logic and Computer Design*, Mano, Pearson.
4. Leach, Malvino, Saha, *Digital Principles and Applications*, TMH.
5. Goel, *Computer Fundamentals*, Pearson.

Digital Electronics and IC Fabrication C Lab Practical: 30 Lectures

	Course Outcome
1.	Implement the basic digital circuits using universal logic gates
2.	Experiment with the ICs to solve problems related to Digital logic circuits
3.	Design and test the combinational circuits, and code conversion methods
4.	Compare various synchronous and asynchronous sequential circuits.

1. Realization of different basic gates by NAND and NOR gates.

2. Realization of any Boolean function.

3. Implement a decoder, encoder, multiplexer, demultiplexer.

4. Implement half adder, full adder.

5. Implement half subtractor, full subtractor.

6. Implement a 1-bit and 2-bit magnitude comparators.

7. Implement S-R flip flop, D flip flop, J-K flip flop, Master-Slave J-K flip flop.

8. Implement counters.

9. Simulation of basic logic circuits using VHDL/Verilog/Logisim.

BCAHMJ201: Data Structures **Theory: 40 Lectures**

	Course Outcome
1.	Define different operations on data structure such as insertion, deletion, merging using arrays
2.	Demonstrate implementation of stacks and queues: insertion, deletion of elements
3.	Construction and implementation of linked lists: inserting, deleting, and inverting a linked list. Analyze implementation of stacks & queues using linked lists,
4.	Design recursive algorithms for different some recursive and non-recursive problems.
5.	Understand the binary tree data structure, its memory representations
6.	Identify the unique binary tree from given any two traversals of a binary tree
7.	Ability to understand different searching algorithms and their applicability
8.	Ability to understand working principle of different sorting techniques
9.	construct a binary search tree (BST), insertion and deletion in BST. Understand the need of balanced search tree
10.	Construction of different hash functions

Unit 1. **Data Structure Introduction** **(4 Lectures)**

Data structure definition and its need, linear and non-linear data structures, writing algorithms.

Unit 2. **Arrays** **(3 Lectures)**

Single and Multi-dimensional Arrays, Advantage and disadvantage of array, Sparse Matrices.

Unit 3**. Stacks** **(5 Lectures)**

Implementing single / multiple stack(s) in an Array; stack top pointer; PUSH, POP operation, overflow and underflow, Prefix, Infix and Postfix expressions, Applications of stack in recursion; Solution of tower of Hanoi.

Unit 4. **Queue** **(4 Lectures)**

Implement queue in array, front and rear pointers, insert and delete, overflow and underflow Array and Linked representation of Queue, Deque, Priority Queue.

Unit 5. **Searching and Sorting** **(8 Lectures)**

Linear search, binary search, sorting: concept of internal and external sorting, in place sorting, stable sort; selection sort, bubble sort, insertion sort, shell sort; Comparison of these sorting algorithms.

Unit 6. **Linked List** **(5 Lectures)**

Singly, Doubly and Circular Lists (Array and Linked representation); insertion and deletion in linked list, linked list reversal.

Unit 7. **Binary Tree** **(8 Lectures)**

Binary tree: root node, left and right child, leaf node, internal and external node; height and depth of a node, complete binary tree, number of unlabeled trees with k nodes, tree traversal: pre-order, in-order, post-order; binary search tree (BST): BST property, in-order successor and predecessor of a node in BST, insertion and deletion of node in BST; height-balance tree: AVL tree.

Unit 8. **Hashing** **(3 Lectures)**

Hash table, keys, different hash functions, collision and resolving the collision; chaining; open addressing: probe sequence, linear probing, quadratic probing, double hashing.

Recommended Books:

1. Horowitz, Sahni, Mehta, *Fundamentals of Data Structures in C++*, University Press, 2011.
2. Mark Allen Weiss, *Data Structures and Algorithms Analysis in Java*, Pearson Education, 3rd Edition, 2013.
3. Tenenbaum, Augenstein, Langsam, *Data Structures Using C and C++*, 2nd Edition, PHI, 2009.
4. Cormen, Leiserson, Rivest, Stein, *Introduction to Algorithms*, PHI, 3rd Edition 2009
5. Kruse, *Data Structures and Program Design in C++*, Pearson, 1999.
6. Goodrich, Tamassia, *Data Structures and Algorithms Analysis in Java*, 4th Edition, Wiley, 2013.

Data Structures Lab Practical: 40 Lectures

	Course Outcome
1.	Ability to implement stack and queue with array data structure.
2.	Ability to design stack from queue and vice versa
3.	Implement multiple stacks in a single array
4.	Implement a linked list using self-referential structure and perform basic operations such as insertion, deletion, traversal, reversal etc.
5.	Implement a binary search tree
6.	Design and implement of different searching and sorting algorithms.

1. Write a program (WAP) to search an element from a list. Give user the option to perform Linear or Binary search.
2. to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.
3. Implement Linked List. Include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists.

4. Implement Circular Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list.

5. Perform Stack operations using Array implementation.
6. Perform Queues operations using Circular Array implementation.
7. WAP to scan a polynomial using linked list and add two polynomials.
8. WAP to create a Binary Search Tree and include following operations in tree:
    a. (Recursive and Iterative Implementation)
    b. Deletion by copying
    c. Deletion by Merging
    d. Search a no. in BST
    e. Display its preorder, postorder and inorder traversals Recursively
    f. Display its preorder, postorder and inorder traversals Iteratively
    g. Display its level-by-level traversals
    h. Count the non-leaf nodes and leaf nodes
    I. Display height of tree
    j. Create a mirror image of tree
    k. Check whether two BSTs are equal or not
9. WAP to convert the Sparse Matrix into non-zero form and vice-versa.
10. WAP to implement Diagonal Matrix using one-dimensional array.
11. WAP to implement Lower Triangular Matrix using one-dimensional array.
12. WAP to implement Upper Triangular Matrix using one-dimensional array.
13. WAP to implement Symmetric Matrix using one-dimensional array.

BCASEC02: Programming in Python Theory: 30 Lectures

	Course Outcome
1.	Learn syntax of Python
2.	Write simple programs in Python for solving simple computation problems
3.	Learn the rich set of libraries
4.	Process strings and lists

Unit 1. Overview of Python Programming: Structure of a Python Program, Elements of Python

Introduction to Python: Python Interpreter, Python shell, Indentation. Atoms, Identifiers and keywords, Literals, Strings, Operators (Arithmetic operator, Relational operator, Logical or Boolean operator, Assignment, Operator, Ternary operator, Bit wise operator, Increment or Decrement operator). (4 Lectures)

Unit 2. Creating Python Programs: Input and Output Statements, Control statements (Branching, Looping, Conditional Statement, Exit function, Difference between break, continue and pass). Nested conditionals, Multiple assignment, The while statement, Tables, Two-dimensional tables. (6 Lectures)

Unit 3. Function: Defining Functions, default arguments, The return statement, Recursion, Stack diagrams for recursive functions. Methods. Lambda function, Regular expression. (4 Lectures)

Unit 4. Strings, List, Tuple, Sets and Dictionary: String as a compound data type, Length, Traversal, String slices, String comparison, find function. List values, Accessing elements, List length, List membership, Lists and for loops, List operations, List deletion. Cloning lists, Nested lists. (5 Lectures)

Unit 5: Errors and Exceptions. (2 Lectures)

Unit 6. Python File Operations: Reading files, Writing files in python, Understanding read and write functions: read(), readline(), readlines(), write() and writelines(). Manipulating file pointer using seek. (3 Lectures)

Unit 7: Python packages: Simple programs using the built-in functions of packages matplotlib, numpy, pandas etc. (3 Lectures)

Unit 8: GUI Programming: Tkinter introduction, Tkinter and Python Programming, Tk Widgets, Tkinter examples. Python programming with IDE. (3 Lectures)

Recommended Books:

1. T. Budd, *Exploring Python*, TMH, 1st Ed, 2011

2. Chun, *Core Python Programming*, Pearson 20016.

3. Charles Dierbach, “Introduction to Computer Science using Python”, Wiley, 2015

4. David Beazley, Brian Jones., “Python Cookbook”, Third Edition, Orelly Publication, 2013

Programming in Python Lab **Practical: 30 Lecture**

	Course Outcome
1.	Write simple programs for solving given computation problems
2.	Design user defined functions
3.	Implement some data structures using Python
4.	Develop some string and list processing functions

1. Using for loop, print a table of Celsius/Fahrenheit equivalences. Let c be the Celsius temperatures ranging from 0 to 100, for each value of c, print the corresponding Fahrenheit temperature.

2. Using while loop, produce a table of sins, cosines and tangents. Make a variable x in range from 0 to 10 in steps of 0.2. For each value of x, print the value of sin(x), cos(x) and tan(x).

3. Write a program that reads an integer value and prints —leap year or —not a leap year.

4. Write a program that takes a positive integer n and then produces n lines of output shown as follows.

For example enter a size: 5

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

5. Write a function that takes an integer ‘n‘ as input and calculates the value of 1 + 1/1! + 1/2!

+ 1/3! + ... + 1/n!

6. Write a function that takes an integer input and calculates the factorial of that number.

7. Write a function that takes a string input and checks if it’s a palindrome or not.

8. Write a list function to convert a string into a list, as in list (‘abc‘) gives [a, b, c].

9. Write a program to generate Fibonacci series.

10. Write a program to check whether the input number is even or odd.

11. Write a program to compare N numbers and print the largest one.

12. Write a program to print factors of a given number.

13. Write a method to calculate GCD of two numbers.

14. Write a program to implement linear search on lists.

15. Write a program to sort a given list.

16. Define a function F as $F_n = F_{n-1} + F_{n-2}$. Write a Python program which accepts a value for N (where N > 0) as input and pass this value to the function. Display suitable error message if the condition for input value is not followed.

17. Develop a python program to convert binary to decimal, octal to hexadecimal using functions.

18. Write a Python program that accepts a sentence and find the number of words, digits, uppercase letters and lowercase letters.

19. Write a function called isphonevalid() to recognize a pattern +91-33-2575 851 with / without using regular expression.

20. Write a function called isphonevalid() to recognize a pattern +91-94345 12345 using regular expression.

21. Write python programs to work with arrays and 2-D arrays using *numpy*.

22. Write python programs to work with series and data frames.

23. Develop a python program that could search the text in a file for phone numbers (+919434123456) and email addresses (rnlkwc@gmail.com).

24. Write a python program to accept a file name from the user to display the first N line of the file.

25. Write a program to input roll numbers and their names of students of your class and store them in the dictionary as the key-value pair and then insert / delete a student's record.

26. Write a python program to design a simple GUI page using Tkinter.

BCAMJ301T: Object-Oriented Programming **Theory: 40 Lectures**

	Course Outcome
1.	Define different operations on data structure such as insertion, deletion, merging using arrays
2.	Demonstrate implementation of stacks and queues: insertion, deletion of elements
3.	Construction and implementation of linked lists: inserting, deleting, and inverting a linked list. Analyze implementation of stacks & queues using linked lists,
4.	Design recursive algorithms for different some recursive and non-recursive problems.
5.	Understand the binary tree data structure, its memory representations
6.	Identify the unique binary tree from given any two traversals of a binary tree
7.	Ability to understand different searching algorithms and their applicability
8.	Ability to understand working principle of different sorting techniques
9.	construct a binary search tree (BST), insertion and deletion in BST. Understand the need of balanced search tree
10.	Construction of different hash functions

Unit 1: **Introduction to OOP** **(5 Lectures)**

Overview of Object-Orientation Programming paradigm and its advantages. Features of OOP: class, object, member variable and function, Class Constructors, Method Overloading, encapsulation, inheritance, polymorphism.

Unit 2: **OOP Languages: C++, Java** **(3 Lectures)**

Java Architecture and Features, Understanding the semantic and syntax differences between C++ and Java, C++ Header Files: iostream, cmath, cstring, cstdlib etc., Java packages. Console I/O in C++: cin, cout . Simple I/O using System.out and the Scanner class in Java. Built-in Java packages**:** java.lang, java.io, java.util etc.

Unit 3: **Programming Basics** **(5 Lectures)**

Declaring, Defining and Initializing Variables, Scope of Variables, Using Named Constants, Keywords, Data Types, Casting of Data Types, Comment line(s) in programs, Operators, scope resolution operator, Expressions, Conditional Statements, Iterative Statements using loop , Use of break and continue in loops, Using Nested Statements (Conditional as well as Iterative)

Unit 4: **Arrays, Strings and Methods** **(8 Lectures)**

Call by Reference, Friend function, Creating & Using Arrays (One Dimension and Multi-dimensional), Referencing Arrays Dynamically, Java Strings: The Java String class, Creating & Using String Objects, Manipulating Strings, String Immutability & Equality, Passing Strings To & From Methods, String Buffer Classes. Byte and Character streams, Reading/Writing from console and files, Garbage Collection.

Unit 5. **Inheritance, Interfaces, Packages, Enumerations, Autoboxing and Metadata** **(10 lectures)**

Inheritance: (Single Level and Multilevel, Method Overriding, Dynamic Method Dispatch, Abstract Classes), Interfaces and Packages, Extending interfaces and packages, Package and Class Visibility, Using Standard Java Packages (util, lang, io, net), Wrapper Classes, Autoboxing/Unboxing, Enumerations and Metadata.

Unit 6: **Exception Handling, Threading** **(5 Lectures)**

Exception types, uncaught exceptions, throw, built-in exceptions, Creating your own exceptions; Multi-threading: The Thread class and Runnable interface, creating single and multiple threads, Thread prioritization, synchronization and communication, suspending/resuming threads.

Unit 7: **Applets and Event Handling** **(4 Lectures)**

Java Applets: Introduction to Applets, Writing Java Applets, Working with Graphics, Incorporating Images & Sounds. Event Handling Mechanisms, Listener Interfaces, Adapter and Inner Classes. The design and Implementation of GUIs using the AWT controls.

Recommended Books:

1. Ken Arnold, James Gosling, David Homes, "The Java Programming Language", 4th Edition, 2005.
2. Cay S. Horstmann, Gary Cornell, "Core Java Volume 1 Fundamentals, Pearson.
3. Bruce Eckel, "Thinking in Java", 3rd Edition, PHI, 2002.
4. E. Balagurusamy, "Programming with Java", 4th Edition, McGraw Hill.2009.
5. Herbert Schildt, *C++: The Complete Reference*, 4th Edition, McGraw Hill.
6. Stroustrup, *The C++ Programming Language*, 4th Edition, Pearson.
7. Lafore, *Object-Oriented Programming in C++*, 4th Edition, Pearson.\
8. Balagurusamy, *Object Oriented Programming with C++*, TMH.

BCAMJ301P Object-Oriented Programming Lab using Java **Practical: 40 Lectures**

	Course Outcome
1.	Interpret the principles of OOP using programming syntaxes of JAVA programming language by analyzing the problems.
2.	Identify the requirements to the solution of complex engineering problems by proper analysis of classes with their relationships and interpretation of data/objects.
3.	Construct computer programs to implement the major OOP concepts related to Class & Object, Polymorphism, Inheritance, Interface, Exception Handling, Multi-processing (Thread), etc using JAVA coding ethics by making use of modern tools like Notepad++, Netbeans or Eclipse IDE.
4.	Develop Graphical User Interfaces using Applet, Swing, Layout manager, Jbutton class with action listeners, etc.
5.	Build small OOP based applications working individually or in a team with proper documentations following the professional OOP based engineering solution techniques.
6.	Determine the need for different OOP components hands-on to produce huge distributed data driven software from the implementation point of view to contribute to lifelong learning.

1. To find the sum of any number of integers entered as command line arguments
2. To find the factorial of a given number
3. To learn use of single dimensional array by defining the array dynamically.
4. To learn use of length in case of a two dimensional array
5. To convert a decimal to binary number
6. To check if a number is prime or not, by taking the number as input from the keyboard
7. To find the sum of any number of integers interactively, i.e., entering every number from the keyboard, whereas the total number of integers is given as a command line argument
8. Write a program that show working of different functions of String and StringBufferclasss like setCharAt(, setLength(), append(), insert(), concat()and equals().
9. Write a program to create a ―distance‖ class with methods where distance is computed in terms of feet and inches, how to create objects of a class and to see the use of this pointer
10. Modify the ―distance‖ class by creating constructor for assigning values (feet and inches) to the distance object. Create another object and assign second object as reference variable to another object reference variable. Further create a third object which is a clone of the first object.
11. Write a program to show that during function overloading, if no matching argument is found, then java will apply automatic type conversions (from lower to higher data type)
12. Write a program to show the difference between public and private access specifiers. The program should also show that primitive data types are passed by value and objects are passed by reference and to learn use of final keyword
13. Write a program to show the use of static functions and to pass variable length arguments in a function.
14. Write a program to demonstrate the concept of boxing and unboxing.
15. Create a multi-file program where in one file a string message is taken as input from the user and the function to display the message on the screen is given in another file (make use of Scanner package in this program).
16. Write a program to create a multilevel package and also creates a reusable class to generate Fibonacci series, where the function to generate Fibonacci series is given in a different file belonging to the same package.
17. Write a program that creates illustrates different levels of protection in classes/subclasses belonging to same package or different packages
18. Write a program ―*DivideByZero* that takes two numbers a and b as input, computes a/b, and

invokes Arithmetic Exception to generate a message when the denominator is zero.

19. Write a program to show the use of nested try statements that emphasizes the sequence of checking for catch handler statements.
20. Write a program to create your own exception types to handle situation specific to your application (*Hint*: Define a subclass of Exception which itself is a subclass ofThrowable).
21. Write a program to demonstrate priorities among multiple threads.
22. Write a program to demonstrate multithread communication by implementing synchronization among threads (Hint: you can implement a simple producer and consumer problem).

BCAMJ302T: Computer Architecture and Organisation: Theory: 60 Lectures

	Course Outcome
1.	Demonstrate sufficient knowledge and understanding of data representation, and experiment with basic arithmetic operations.
2.	Analyze and model various functional units of CPU such as ALU, control unit and register file.
3.	Organize the memory hierarchy and design a memory of any type.
4.	Explain the instruction set architecture, instruction formats and instruction cycle.
5.	Outline various modes of I/O operations and summarize working principles of I/O interface circuits.
6.	Explain the pipelining technique and its related issues.

Unit 1: **Basic Computer Organization and Design** **(10 lectures)**

Computer Organization vs Computer Architecture, von Neumann and Harvard architecture, Computer registers: program counter, instruction register, accumulator, stack pointer, status register, GPRs, bus system, Interconnection Structures, Bus Interconnection design of basic computer.

Unit 2: **Central Processing Unit** **(12 lectures)**

CPU registers: accumulator, program counter, instruction register, stack pointer. Length of different CPU registers for a given word length and memory size, instruction set, Instruction formats, addressing modes, instruction codes, instruction execution cycle: opcode fetch, decode, read operands, execute, write result. Sequential execution and pipelined execution, speedup, arithmetic and logical micro-operations, stack organization. Design of a simple 1-bit ALU, bit-sliced ALU design, Control Unit: hardwired and micro programmed control. RISC and CISC architectures, parallel architecture.

Unit 3: **Memory Organization**

(12 lectures)

Principle of locality: temporal and spatial locality, need of memory hierarchy, von Neumann bottleneck. Cache memory: hit, miss, hit ratio, effective memory access time, write back cache, write through cache, cache memory organizations: direct mapped, fully associative, set associative, Secondary memory: seek time, latency time, virtual memory.

Unit 4: **Input-Output Organization** **(6 lectures)**

Parallel and sequential I/O, synchronous and asynchronous I/O, baud rate, interrupt, interrupt service routine, Programmed I/O, Interrupt-Driven I/O, daisy-chaining, polling, Direct Memory Access: burst-transfer mode and cycle-stealing mode.

Recommended Books:

1. M. Mano, *Computer System Architecture*, Pearson Education.

2. W. Stallings, *Computer Organization and Architecture Designing for Performance*, PHI.

3. Patterson, Hennessy, *Computer Organization and Design*, MK Publication.

4. M. Mano , *Digital Design*, Pearson Education.

5. Carl Hamacher, *Computer Organization*, TMH.

6. Pal Chaudhury, *Computer Organization and Design*, PHI.

7. Hayes, *Computer Architecture and Organization*, TMH.

8. Sajjan G Shiva, *Computer Organization Design and Architecture*, CRC Press.

BCAMJ302P: 8085 Assembly Language Programming **Practical: 40 Lecture**

	Course Outcome
1.	Learn 8085 software architecture
2.	Learn and apply 8085 instructions for different tasks
3.	Ability to write 8085 ALP
4.	Ability to write 8085 ALP using subroutine
5.	Ability to use in-built subroutines available in kit

1. Study of 8085 microprocessor software architecture and 8085 instructions set.
2. Familiarization with 8085 register level architecture and trainer kit and TASM components, including the memory map. Familiarization with the process of storing and viewing the contents of memory as well as registers.

3. Study of prewritten programs on trainer kit using the basic instruction set (data transfer, Load/Store, Arithmetic, Logical) Assignments based on above.
4. Programming using kit/simulator for i) table look up ii) Copying a block of memory iii) Shifting a block of memory. iv) Packing and unpacking of BCD numbers, v) binary to gray code converter, vi) addition of 16-bit numbers, vii) subtraction of 16-bit numbers, vii) minimum of numbers, viii) swapping of lower nibble with higher nibble, ix) masking and setting selective bits, x) 8-bit multiplication using subroutine, xi) sorting using subroutine, xii) use of in-built subroutines like UPDDT, UPDAD etc.

Recommended Books:

1. R. Gaonkar, 8085 Microprocessor, Penram International.

2. S. Mandal, 8085 Microprocessor Programming, TMH.

BCASEC03T: Web Technologies Theory: 30 Lectures

	Course Outcome
1.	To examine and evaluate the need for secured web application development with client-side, server-side scripting languages.

2.	To construct web programs using the web languages--HTML, XML, JavaScript, Applet, Perl, etc.
3.	To design and develop small interactive websites using modern tools following the professional web based engineering solutions, ethics and management techniques.
4.	To determine and combine the advanced technologies like network security, multimedia applications, search engine, web crawler, etc with the websites.

Unit 1: Introduction to WWW- Protocols and programs, secure connections, application and development tools, the web browser, What is server, choices, setting up UNIX and Linux web servers, Logging users, dynamic IP Web Design: Web site design principles, planning the site and navigation, (6 Lectures)

Unit 2: Introduction to HTML- The development process, Html tags and simple HTML forms, web site structure Introduction to XHTML: XML, Move to XHTML, Meta tags, Character entities, frames and frame sets, inside browser. HTML Tags: Table, list, link etc (10 Lectures)

Unit 3: Cascading Style sheets- Need for CSS, introduction to CSS, basic syntax and structure, using CSS, background images, colours and properties, manipulating texts, using fonts, borders and boxes, margins, padding lists, positioning using CSS. (8 Lectures)

Unit 4: Javascript- Client side scripting, What is Javascript, How to develop Javascript, simple Javascript, variables, functions, conditions, loops and repetition. JavaScript alert, prompt and confirm. Objects in JavaScript, Access/Manipulate web browser elements using DOM Structure, forms and validations, JavaScript events, Basics of jQuery, jQuery syntaxes, jQuery selectors, events, effects, Access / Manipulate web browser elements using jQuery. (8 Lectures)

Unit 5: Introduction to PHP and its syntax, combining PHP and HTML, understanding PHP code blocks like Arrays, Strings, Functions, looping and branching, file handling, processing forms on server side, cookies and sessions. Introduction to PHP MyAdmin, connection to MySQL server from PHP, execution of MySQL queries from PHP, receiving data from database server and processing it on webserver using PHP. (8 Lectures)

Recommended Books:

1. Ivan Bayross, *Web Enabled Commercial Application Development Using HTML, DHTML, Javascript, Perl CGI*, BPB Publications, 2009.
2. Hans Bergsten, *Java Server Pages*, O'Reilly , Third Edition, 2003.
3. Knuckles, *Web Applications: Concepts and Real World Design*, Wiley-India.
4. P.J. Deitel & H.M. Deitel, *Internet and World Wide Web How to program*, Pearson.
5. Julie C. Meloni, *HTML, CSS and JavaScript All in One*, Sams Teach Yourself, 2nd Edition.
6. Paul Wellens, *Practical Web Development*, PACKT Publication.

BCASEC03P : Web Technologies Lab **Practical: 30 Lectures**

	Course Outcome
1.	Learn finer details of website development process
2.	Design simple as well as moderate websites
3.	Apply knowledge of HTML, CSS, Javascript and PHP in designing dynamic website

Practical on HTML

1. Design web pages for your college containing a description of the courses, departments, faculties, library etc, use href, list tags.

2. Create your class time table using table tag.
3. Create user Student feedback form (use textbox, text area , checkbox, radio, button, select box etc.)
4. Create a web page using frame. Divide the page into two parts with Navigation links on left hand side of page (width=20%) and content page on right hand side of page (width = 80%). On clicking the navigation Links corresponding content must be shown on the right hand side.
5. Write html code to develop a webpage having two frames that divide the webpage into two equal rows and then divide the row into equal columns fill each frame with a different background color.
6. Create your resume using HTML tags also experiment with colors, text , link , size and also other tags you studied.

Practical on CSS

7. Design a web page of your home town with an attractive background color, text, color, an Image, font etc. (use internal CSS).
8. Use Inline CSS to format your resume that you created.
9. Use External CSS to format your class timetable as you created.
10. Use External, Internal, and Inline CSS to format college web page that you created.

Practical on JavaScript

11. Develop a JavaScript to display today's date.
12. Develop simple calculator for addition, subtraction, multiplication and division operation using JavaScript
13. Create HTML Page with JavaScript which takes Integer number as input and tells whether the number is ODD or EVEN.
14. Create HTML Page that contains form with fields Name, Email, Mobile No, Gender , Favourite Colour and a button now write a JavaScript code to combine and display the information in textbox when the button is clicked.

Practicals on PHP

15. Server side scripts and validation arrays for a simple log-in page of website.

BCAMJ401T: Operating Systems **Theory: 40 Lectures**

	Course Outcome
1.	Illustrate the resource-management by the Operating System (OS) and describe the basic principles used in the design of modern OS
2.	Apply various CPU scheduling algorithms for any given problem and outline the needs and applications of process synchronization
3.	Identify the issues in deadlock in terms of avoiding, preventing and recovering the same
4.	Design solutions using semaphore variables to solve different mutual exclusion problems
5.	Elaborate the different schemes used in memory management including paging and segmentation
6.	Analyze various file and disk management strategies
7.	Justify the issues in I/O management and security

1. Introduction (6 Lectures)

Basic OS functions, resource abstraction, types of operating systems–multiprogramming systems, batch systems, time sharing systems; operating systems for personal computers & workstations, process control & real time systems.

2. Operating System Organization (6 Lectures)
Processor and user modes, kernels, system calls and system programs.

3. Process Management (10 Lectures)
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; concurrent and processes, critical section, semaphores, methods for inter- process communication; deadlocks.

4. Memory Management (10 Lectures)
Physical and virtual address space; memory allocation strategies –fixed and variable partitions, paging, segmentation, virtual memory

5. File and I/O Management (4 Lectures)
Directory structure, file operations, file allocation methods, device management.

6. Protection and Security (4 Lectures)
Policy mechanism, Authentication, Internal access Authorization.

Recommended Books:

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

BCAMJ401P: Operating Systems Lab — Practical: 40 Lectures

	Course Outcome
1.	Execute the Linux commands to perform the basic operations related to process and system.
2.	Know different system calls in Linux and their usage
3.	Demonstrate the execution of the programs like creating new process, creating orphan process and zombie process based on child-parent relationship
4.	Apply the knowledge of semaphore and thread
5.	Implement different CPU scheduling algorithms
6.	Implement some basic memory allocation strategies

1. WRITE A C/C++ PROGRAM (WAP) (using *fork()* and/or *exec()* commands) where parent and child execute:
   a) same program, same code.
   b) same program, different code.
   c) before terminating, the parent waits for the child to finish its task.
2. WAP to report behavior of Linux kernel including kernel version, CPU type and model. (CPU information)
3. WAP to report behavior of Linux kernel including information on configured memory, amount of free and used memory. (memory information)
4. WAP to print file details including owner access permissions, file access time, where file name is given as argument.
5. WAP to copy files using system calls.
6. WAP to implement FCFS scheduling algorithm.
7. WAP to implement Round Robin scheduling algorithm.
8. WAP to implement SJF scheduling algorithm.
9. WAP to implement non-preemptive priority-based scheduling algorithm.

10. WAP to implement preemptive priority-based scheduling algorithm.
11. WAP to implement SRJF scheduling algorithm.
12. WAP to calculate sum of n numbers using *thread* library.
13. WAP to implement first-fit, best-fit and worst-fit allocation strategies.

BCAMJ402T: Design and Analysis of Algorithms Theory: 40 Lectures

	Course Outcome
1.	Understand asymptotic notations, determine time complexity of different algorithms from their recurrence relation.
2.	Comprehend different algorithm design paradigms
3.	Analyze use of divide-and-conquer strategy in merge sort, quick sort
4.	Appreciate the use of greedy technique to obtain optimum global solution for some problems
5.	Apply backtracking method to solve n-queens problems
6.	Understand the applicability of dynamic programming for some problems
7.	Comprehend the graph data structure, its traversal, and solution of minimum spanning tree

1. Introduction (6 Lectures)
Basic Design and Analysis techniques of Algorithms, Correctness of Algorithm. Time Complexity, Master's Theorem in solving recurrence relation.

2. Algorithm Design Techniques (5 Lectures)
Iterative techniques, Divide and Conquer, Dynamic Programming, Greedy Algorithms.

3. Sorting and Searching Techniques (15 Lectures)
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;

4. **Red-Black Tree**. (5 Lectures)

5. **Graph Algorithms:** Breadth First Search, Depth First Search and its Applications, Minimum Spanning Trees. (5 Lectures)

6. String Processing: (4 Lectures)
String Matching, KMP Technique

Recommended Books:

1. Cormen, Leiserson, Rivest, Stein, *Introduction to Algorithms*, PHI, 3rd Edition 2009.

2. Horowitz, Sahni, Rajasekaran, *Computer Algorithm*, 2nd Edition, Orient Blackswan Publication.

3. Sarabasse, Gelder, *Computer Algorithm – Introduction to Design and Analysis*, 3rd Edition, Pearson Education.

4. Kleinberg, Tardos, *Algorithm Design*, Pearson Education.

5. Basu, *Design Methods and Analysis of Algorithms*, PHI.

BCAHMJ402P: Design and Analysis of Algorithms Lab **Practical: 40 Lectures**

	Course Outcome
1.	Understand divide and conquer approach and to implement binary search, merge sort, quick sort
2.	Analysis of working principle of heap sort and radix sort by implementing them
3.	Implement graph traversal algorithms and analyse its output
4.	Apply backtracking method to solve n-queens problems.
5.	Design and Develop program to solve minimum spanning tree problem

1. i. Implement Insertion Sort (The program should report the number of comparisons)
   ii. Implement Merge Sort(The program should report the number of comparisons)
2. Implement Heap Sort(The program should report the number of comparisons)
3. Implement Randomized Quick sort (The program should report the number of comparisons)
4. Implement Radix Sort
5. Create a Red-Black Tree and perform following operations on it:
   i. Insert a node
   ii. Delete a node
   iii. Search for a number & also report the color of the node containing this number.
6. Write a program to determine the LCS of two given sequences
7. Implement Breadth-First Search in a graph
8. Implement Depth-First Search in a graph
9. Write a program to determine the minimum spanning tree of a graph

For the algorithms at S.No 1 to 3 test run the algorithm on 100 different inputs of sizes varying from 30 to 1000. Count the number of comparisons and draw the graph. Compare it with a graph of $n\log_2 n$.

BCAMJ403T: Software Engineering **Theory: 40 lectures**

	Course Outcome
1.	Explain the principles of software engineering in the context of social, ethical, legal, economic and environmental concerns by building applicable solutions.
2.	Identify and classify the customer requirements to the solution of complex engineering problems by proper analysis and interpretation of data and processes.
3.	Estimate software matrices like size, effort and cost, software reliability and quality, etc and apply project management techniques to maximize the productivity.
4.	Design various components of software using DFD, ERD, Modularization, Use-case diagram, Class diagram, Sequence diagram, etc. following the professional software design guidelines.
5.	Develop and Test software products following standard coding and testing guidelines.
6.	Asses the utility of various components of software development process and to combine them to produce different types of software to adapt in the software industries in future.

Introduction (10 Lectures)
The Evolving Role of Software, Software Characteristics, Changing Nature of Software, Software Engineering as a Layered Technology, Software Process Framework, Framework and Umbrella Activities, Process, Software Development Life Cycle. Waterfall model of SDLC.

Requirement Analysis (8 Lectures)
Software Requirement Analysis, Initiating Requirement Engineering Process, Requirement Analysis and Modeling Techniques, Flow Oriented Modeling, Need for SRS, Characteristics and Components of SRS.

Software Project Management (6 Lectures)
Estimation in Project Planning Process, Project cost estimation metrics, COCOMO model, Project Scheduling. Project risk.

Quality Management (4 Lectures)
Quality Concepts, Software Quality Assurance, Software Reviews, Metrics for Process and Projects.

Design Engineering (6 Lectures)
Design Concepts, Architectural Design Elements, Software Architecture, Data Design at the Architectural Level and Component Level, Mapping of Data Flow into Software Architecture, Modeling Component Level Design.

Testing Strategies & Tactics (6 Lectures)

Software Testing Fundamentals, Strategic Approach to Software Testing, Test Strategies for Conventional Software, Validation Testing, System testing, Black-Box Testing, White-Box Testing and their type, Basis Path Testing.

Recommended Books:

1 .R.S. Pressman, Software Engineering: A Practitioner‘s Approach (7th Edition), McGraw-Hill, 2009.
2 .P. Jalote, An Integrated Approach to Software Engineering (2ndEdition), Narosa Publishing House, 2003.
3. I. Sommerville, Software Engineering (8th edition), Addison Wesley, 2006.
4. D. Bell, Software Engineering for Students (4th Edition), Addison-Wesley, 2005.
5. R. Mall, Fundamentals of Software Engineering (2nd Edition), Prentice-Hall of India, 2004.

BCAMJ403P: Software Engineering Lab **Practical: 40 Lecture**

	Course Outcome
1.	Identify and classify the customer requirements for the solution of complex engineering prob-lems by proper analysis and interpretation of data and processes supported by standard docu-mentation.
2.	Analyzethe software processes by mapping requirements in to Use case diagrams/ Data Flow Diagrams and Entity Relationship Diagrams for given case studies
3.	Experiment with modern tools like Rational rose, Smartdraw, Erdraw, etc. to design dynamic behaviour of software with modular programming, class diagrams, sequence diagrams, etc.
4.	Estimate software metric like size, effort and cost , software reliability and quality, etc and pla development schedule using PERT and GNATT charts
5.	Design the Test cases and the Test suits for the given case studies using Black box and White box techniques
6.	Determine and evaluate the various components of software development process practically and to combine them to produce different types of software to adapt in the software industrie in future

S. No	Practical Title
1.	Problem Statement, Process Model
2.	Requirement Analysis: Creating a Data Flow Data Dictionary, Use Cases Project Management:
3	Computing FP Effort • Schedule, Risk Table, Timeline chart
4.	Design Engineering: Architectural Design • Data Design, Component Level Design
5.	Testing: Basis Path Testing

Sample Projects:

1. Criminal Record Management: Implement a criminal record management system for jailers, police officers and CBI officers

2. DTC Route Information: Online information about the bus routes and their frequency and fares

3. Car Pooling: To maintain a web based intranet application that enables the corporate employees within an organization to avail the facility of carpooling effectively.

4. Patient Appointment and Prescription Management System

5. Organized Retail Shopping Management Software

6. Online Hotel Reservation Service System

7. Examination and Result computation system

8. Automatic Internal Assessment System

9. Parking Allocation System

10. Wholesale Management System

BCAMJ501T : Database Management System Theory: 40 Lectures

	Course Outcome
1.	Define and understand the fundamentals of Data base management System and traditional file system.
2.	Understand and explain the concepts of relational database management system.
3.	Make use of the tools to implement Entity Relationship diagrams.
4.	Utilize and take part in the normalization of the real world database to remove redundancies and able to apply the conversion of one Normal Form to Higher Normal Form.
5.	Elaborate the importance and rule on database management system concepts to minimize conflict in concurrent transactions.
6.	Discuss the importance of Database management system for storage of data in various formats and able to judge the environmental, societal and market issues specific to software development.

1. Introduction (4 Lectures)
Characteristics of database approach, data models, database system architecture and data independence.

2. Entity Relationship(ER) Modeling (6 Lectures)

Entity types, relationships, constraints.

3. Relation data model (10 Lectures)

Relational model concepts, relational constraints, relational algebra, SQLqueries

4. Database design (10 Lectures)
Mapping ER/EER model to relational database, functional dependencies, Lossless decomposition, Normal forms(upto BCNF).

5. Transaction Processing (4 Lectures)

ACID properties, concurrency control
6. File Structure and Indexing (6 Lectures)
Operations on files, File of Unordered and ordered records, overview of File organizations, Indexing structures for files(Primary, secondary, and clustering index), Multilevel indexing with B and B+ tree

Books Recommended:

1. Elmasri, Navathe, *Fundamentals of Database Systems*, 6th Edition, Pearson Education, 2010.

2. Silberschatz, Korth, Sudarshan, *Database System Concepts*, 6th Edition, TMH.

3. Ramakrishanan, Gehrke, *Database Management Systems*, 3rd Edition, TMH. 2002.

BCAMJ501P : Database Management System Lab Practical: 40 Lectures

	Course Outcome
1.	Outline the underlying concepts of database technologies.
2.	Define and demonstrate DBMS architecture, schema, instance, DDL, DML.
3.	Experiment with SQL to construct and apply to execute database query using SQL DML/DDL commands.
4.	List and test the integrity constraints on a database using a RDBMS and discover relationships.
5.	Explain Programming in PL/SQL with stored procedures, cursors, packages.
6.	Compose and improve/solve the need of DBMS tool for the use of modern software development.

Structured Query Language
1. Creating Database
Creating a Table
Specifying Relational Data Types
Specifying Constraints
Creating Indexes
2. Table and Record Handling

INSERT statement
Using SELECT and INSERT together
DELETE, UPDATE, TRUNCATE statements
DROP, ALTER statements

3. Retrieving Data from Database

The SELECT statement
Using the WHERE clause
Using Logical Operators in the WHERE clause
Using IN, BETWEEN, LIKE, ORDER BY, GROUP BY and HAVING Clause Using Aggregate Functions
Combining Tables Using JOINS
Subqueries

4. Database Management

Creating Views
Creating Column Aliases
Creating Database Users
Using GRANT and REVOKE

5. PL/SQL

BCAMJ502T: Computer Networks Theory: 40 Lectures

	Course Outcome
1.	Explain data communication system, components and the purpose of layered architecture.
2.	Illustrate the functionalities of each layer of OSI and TCP/IP reference model including their associated protocols.
3.	Understand the IP addressing schemes, subnet mask and the address used for broadcast
4.	Understand the routing protocols
5.	Apply the different error detection and correction schemes with examples
6.	Comprehend the different commonly used application layer protocols

1. **Introduction to Computer Networks** (4 Lectures)

Network definition; network topologies; network classifications; network protocol; layered network architecture; overview of OSI reference model; overview of TCP/IP protocol suite.

2. **Data Communication Fundamentals and Techniques** (4 Lectures)

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.

Networks Switching Techniques and Access mechanisms (6 Lectures)

Circuit switching; packet switching- connectionless datagram switching, connection-oriented virtual circuit switching; dial-up modems; digital subscriber line; cable TV for data transfer.

3. **Data Link Layer Functions and Protocol** (6 Lectures)

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.

4. **Multiple Access Protocol and Networks** (4 Lectures)

CSMA/CD protocols; Ethernet LANS; connecting LAN and back-bone networks- repeaters, hubs, switches, bridges, router and gateways;

5. **Networks Layer Functions and Protocols** (6 Lectures)

Routing; routing algorithms; network layer protocol of Internet- IP protocol, Internet control protocols.

6. **Transport Layer Functions and Protocols** (6 Lectures)

Transport services- error and flow control, Connection establishment and release- 3-way handshake;

7. **Overview of Application layer protocol** (4 Lectures)

Overview of DNS protocol; overview of WWW &HTTP protocol.

Recommended Books

1. Kurose, Ross, *Computer Networking*: A Top Down Approach, 7th Edition, Pearson Education.
2. Forouzan, *Data Communications and Networking*, 4th Edition, TMH, 2007.
3. Tanenbaum, *Computer Networks*, 4th Edition, PHI , 2002.
4. Peterson, *Computer Network: A Systems Approach*, Morgan Kauffman Publication.

BCAMJ502P: Computer Networks Lab **Practical: 40 Lectures**

	Course Outcome
1.	Implement program to read an IP address and determine its class
2.	Understand different library functions used for creation of socket, establishment of connections between client and server using TCP and UDP
3.	Implement client-server communication using TCP
4.	Implement client-server communication using UDP
5.	Implement some error detection algorithms
6.	Simulate some routing protocols

1. Simulate Cyclic Redundancy Check (CRC) error detection algorithm for noisy channel.
2. Simulate and implement stop and wait protocol for noisy channel.
3. Simulate and implement distance vector routing algorithm
4. Simulate and implement Dijkstra algorithm for shortest path routing.
5. WAP for Creation of sockets
6. Write a program to communicate between TCP client & server
7. Write a program to communicate between UDP client & socket
8. Write a socket Program for Echo/Ping/Talk commands

BCAMJ503T: Theory of Computation **Theory: 60 Lectures**

	Course Outcome
1.	Understand different class of grammars and languages
2.	Ability to design DFA for a given regular language
3.	Ability to design pda for a given CFL
4.	Skill to design Turing machine for a given recursively enumerable language

Module-1: Alphabet, strings and languages. Definable languages, countable and uncountable sets, Cantor's theorem and proof that there are uncountably many languages, but only countable of them are definable. (8 Lectures)

Module-2: **Regular languages ad Finite Automata**: deterministic and non-deterministic finite automaton (DFA and NFA). Equivalence of DFA and NFA. Minimization of DFA. Some closure properties of regular languages. Regular expressions, construction of NFA and DFA from a regular expression. Pumping lemma and existence of non-regular languages. Mealy machine, Moore machine. (16 Lectures)

Module-3: **Context-free languages (CFL) and Pushdown Automata**: context-free grammar (CFG), push-down automaton (PDA), equivalence of CFG and PDA. Closure properties of CFLs. Chomsky normal form (CNF). Pumping lemma of CFL and existence of non-context-free languages. Context-sensitive grammar and language. (15 Lectures)

Module-4: **Turing Machine**: Turing machine recognizable or recursively enumerable languages, recursive languages. Variants of Turing machine model and their equivalence. Decision problems of formal languages. Universal Turing machine. Language not recognized by Turing machine and undecidable problems of formal language. Mapping reducibility of decision problems. (15 Lectures)

Module-5: Introduction to Complexity Classes: P, NP, NPC, NP-Hard. (6 Lectures)

References:

1. Michael Sipser, *Introduction to Theory of Computation*, Cengage Learning.

2. Hopcroft, Motwani & Ullman, *Introduction to Automata Theory, Languages, and Computation*, Pearson Education.

3. Z. Kohavi and N.K. Jha, *Switching and Finite Automata Theory*, TMH.

4. Mishra & Chandrasekaran, *Theory of Computer Science, Automata, languages and Computation*, PHI.

5. Lewis and Papadimitrou, *Elements of the theory of Computation*, PHI.

6. Linz Peter, *An Introduction to Formal Languages and Automata*, Narosa.

BCAMJ504T : Artificial Intelligence **Theory: 40 Lectures**

	Course Outcome
1.	Explain the modern tools of Artificial Intelligence (AI) and knowledge representation models.
2.	Determine the domain-specific problems solving methods using AI based algorithms and techniques.
3.	Evaluate knowledge based representation and learning methods for broad areas of state-of-the-art technological growth.
4.	Develop and access some current applications of AI in the fields of Expert Systems, Robotics, Machine Learning and others.
5.	Design the efficient solutions using different AI based schemes, and relate with technological advancement for future learning.

Unit-1. Introduction (8 Lectures)

Introduction to Artificial Intelligence, Background and Applications, Turing Test and Rational Agent approaches to AI, Introduction to Intelligent Agents, their structure, behaviour and environment.

Unit-2. Problem Solving and Searching Techniques: (16 Lectures)

Problem Characteristics, Production Systems, Control Strategies, Breadth First Search, Depth First Search, Hill climbing and its Variations, Heuristics Search Techniques: Best First Search, A* algorithm, Constraint Satisfaction Problem, Means-End Analysis, Introduction to Game Playing, Min-Max and Alpha-Beta pruning algorithms.

Unit-3. Knowledge Representation: (8 Lectures)

Introduction to First Order Predicate Logic, Programming in Logic (PROLOG)

Unit-4. Dealing with Uncertainty and Inconsistencies: (8 Lectures)

Truth Maintenance System, Default Reasoning, Probabilistic Reasoning, Bayesian Probabilistic Inference, Possible World Representations.

Reference Books:

1. Russell & Norvig, Artificial Intelligence -A Modern Approach, Prentice Hall, 3rd edition, 2009.
2. Rich & Knight, Artificial Intelligence – Tata McGraw Hill, 2nd edition, 1991.
3. Ela Kumar, Artificial Intelligence, Wiley.

BCAMJ504P Artificial Intelligence Lab **Practical: 40 Lectures**

	Course Outcome
1.	Learn Prolog programming language
2.	Apply knowledge of Prolog to solve some common computation problems
3.	Apply knowledge of Prolog to accomplish some list operations

List of Practicals:

1. Write a prolog program to calculate the sum of two numbers.
2. Write a prolog program to find the maximum of two numbers.
3. Write a prolog program to calculate the factorial of a given number.
4. Write a prolog program to calculate the nth Fibonacci number.
5. Write a prolog program, insert_nth(item, n, into_list, result) that asserts that result is the list into_list with item inserted as the n'th element into every list at all levels.
6. Write a Prolog program to remove the Nth item from a list.
7. Write a Prolog program, remove-nth(Before, After) that asserts the After list is the Before list with the removal of every n'th item from every list at all levels.
8. Write a Prolog program to implement append for two lists.
9. Write a Prolog program to implement palindrome(List).
10. Write a Prolog program to implement max(X,Y,Max) so that Max is the greater of two numbers X and Y.
11. Write a Prolog program to implement reverse(List,ReversedList) that reverses lists.

BCAMJ601T: Computer Graphics Theory: 40 Lectures

	Course Outcome
1.	Outline computer graphics system, display devices and various application areas of graphics.
2.	Develop scan conversion algorithms for line, circle and ellipse with examples.
3.	Demonstrate and illustrate 2D and 3D transformation operations such as translation, rotation, scaling, etc.
4.	Analyze and model any kind of 3D objects using viewing, clipping and projection techniques.
5.	Apply various curve and surface representation methods such as BSpline, Bezier, etc.
6.	Demonstrate and discuss various hidden surface removal algorithms, and lighting and shading models.

1. Introduction (4 Lectures)
Basic elements of Computer graphics, Applications of Computer Graphics.

2. Graphics Hardware (4 Lectures)

Architecture of Raster and Random scan display devices, input/output devices.

3. Fundamental Techniques in Graphics (12 Lectures)

Raster scan line, circle and ellipse drawing algorithms: DDA, Bresenham, Midpoint; Thick primitives, Boundary filling, Polygon filling, line and polygon clipping algorithms, 2D and 3D Geometric Transformations, 2D and 3D Viewing Transformations (Projections- Parallel and Perspective), Vanishing points.

4. Geometric Modeling (10 Lectures)
Representing curves & Surfaces.

5. Visible Surface determination (6 Lectures)

Hidden surface elimination.

6. Surface rendering (4 Lectures)

Illumination and shading models. Basic color models and Computer Animation.

Books Recommended:

1. J.D. Foley, A.Van Dan, Feiner, Hughes Computer Graphics Principles & Practice, Addison Wesley
2. D. Hearn, Baker: Computer Graphics, PHI
3. D.F. Rogers Procedural Elements for Computer Graphics, McGraw Hill 1997.
4. D.F. Rogers, Adams Mathematical Elements for Computer Graphics, TMH
5. Xiang, Plastock, Schaum's Outline Computer Graphics, TMH

BCAMJ601P: Computer Graphics Lab **Practical: 40 Lectures**

	Course Outcome
1.	Implement drawing lines, circles, certain curves
2.	Transform a given object by rotation, scaling
3.	Clip certain portion of objects
4.	Design a drawing by using simple objects and color different components

1. Write a program to implement DDA line drawing algorithm
2. Write a program to implement Bresenham‘s line drawing algorithm.
3. Write a program to implement Bresenham‘s circle drawing algorithm
4. Write a program to implement mid-point circle drawing algorithm.
5. Write a program to implement Boundary fill algorithm
6. Write a program to clip a line using Cohen and Sutherland line clipping algorithm.
7. Write a program to clip a polygon using Sutherland Hodgeman algorithm.
8. Write a program to apply various 2D transformations on a 2D object (use homogenous coordinates).
9. Write a program to apply various 3D transformations on a 3D object and then apply parallel and perspective projection on it.
10. Write a program to draw Hermite/Bezier curve.

BCAMJ602T: Machine Learning **Theory: 40 Lectures**

	Course Outcome
1.	To Understand a wide variety of learning algorithms.
2.	To understand how to apply a variety of learning algorithms to data using various tools of Machine Learning.
3.	To identify the strengths and weaknesses of many popular machine learning approaches.
4.	To analyze the performance of learning algorithms and model selection
5.	To identify mathematical relationships within and across Machine Learning algorithms and the paradigms of supervised and unsupervised learning.
6.	To apply machine learning techniques in solving complex real world problems.

Unit 1. Basics: Introduction to Machine Learning - Different Forms of Learning, Basics of Probability Theory. (4 Lectures)

Unit 2. Regression Analysis: Linear Regression, Ridge Regression, Bayesian Regression. (4 Lectures)

Unit 3. Classification Methods: Instance-Based Classification, Linear Discriminant Analysis, Logistic Regression, Large Margin Classification, Kernel Methods, Support Vector Machines, Multi-class Classification, Classification and Regression Trees. (8 Lectures)

Unit 4. Neural Networks: Non-linear Hypotheses, Neurons and the Brain, Model Representation, Multi-layer Networks, Back-propagation, Multi-class Discrimination, Training Procedures, Localized Network Structure, Deep Learning. (6 Lectures)

Unit 5. Graphical Models: Hidden Markov Models, Bayesian Networks. (4 Lectures)

Unit 6. Clustering: Partitional Clustering - K-Means, K-Medoids, Hierarchical Clustering - Agglomerative, Divisive, Distance Measures, Density Based Clustering. (6 Lectures)

Unit 7. Dimensionality Reduction: Principal Component Analysis, Independent Component Analysis. (4 Lectures)

Unit 8. Reinforcement Learning: Q-Learning, Temporal Difference Learning. (4 Lectures)

Reference Books:

1. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer.

2. R. O. Duda, P. E. Hart, and D.G. Stork, *Pattern Classification*, Wiley.

3. T. M. Mitchell, *Machine Learning*, McGraw-Hill.

BCAMJ602P: Machine Learning Lab **Practical: 40 Lectures**

	Course Outcome
1.	Familiarity with different tools for machine learning
2.	Learn to find standard datasets
3.	Implement linear regression
4.	Implement different clustering algorithms
5.	Implement different classification algorithms

The programs can be implemented in Python using Jupyter notebook or Google Colab.

Data sets can be taken from standard repositories.

1. Write a program to implement various distance metrics such as Euclidean distance, Manhattan distance, Minkowski distance, Chebychev distance etc.

2. Write a program to construct confusion matrix and determine precision, accuracy, recall etc.

3. Write a program for linear regression in order to fit data points. Select appropriate data set for your experiment and draw graphs.

4. Write a program for polynomial regression.

5. Write a program for logistic regression.

6. Write a program for k-NN algorithm.

7. Write a program for the naive Bayesian classifier for a sample training data set stored as a .CSV file.

8. Write a program for naive Bayesian Classifier.

9. Write a program for clustering using k-means algorithm.

10. Write a program to implement the PCA algorithm. Select appropriate data set for your experiment and draw graphs.

BCAMJ603T: Compiler Design **Theory: 60 Lectures**

	Course Outcome
1.	Understand different phases of compilation
2.	Ability to identify different tokens in a given code
3.	Ability to design top-down and bottom up parse tree for a given input string
4.	Skill to design SLR
5.	Develop the intermediate representation for a given code segment

Module-1: **Introduction:** Overview of the structure of a compiler. Different phases of compilation. Compilation for different targets, front-end and back-end. Idea of assembler, linker and library. (6 Lectures)

Module-2: **Lexical analyzer or scanner**: Tokens and their attributes from the input character stream. Extended regular expressions to specify set of tokens. Synthesis of finite state machine from the specification. *flex*: a lexical analyzer generator. (8 Lectures)

Module-3: **Parser or syntax analyzer**: Specification of context-free structure of a language by CFG, context-sensitive features. Parse tree, syntax tree, different derivations, ambiguous grammar. Transformations of a grammar, removal of left-recursion and left-factoring. Idea of top-down and bottom-up parsing. (12 Lectures)

Module-5: **Top-down parsing**: Computation of First() and Follow(). Recursive descent predictive parser and table driven predictive parser. Error recovery. (8 Lectures)

Module-6: **Bottom-up parsing**: Dotted items, right-most derivation and automata of viable prefixes. Basic structure of shift-reduce parsing. Automaton of LR(0) items, LR(0) and SLR grammar and their parsing. Automaton of LR(1) items, LR(1) grammar, LALR grammar and their parsing. *Bison*: A parser generator. Use of ambiguous operator grammar. Error recovery. (12 Lectures)

Module-7: **Semantic actions and synthesis of attributes**: S and L attributed grammars. Syntax directed definition mars. Syntax directed definitions of different constructs of a programming language. (6 Lectures)

Module-8: **Intermediate representation and Target code generation**: Annotated syntax tree, symbol table, 3-address code. Basic block and control-flow graph. SSA form. Intermediate code generation. Code improvement, Peep hole, within basic block. Notion of data-flow analysis and its purpose. Register allocation. Target code generation. (8 Lectures)

References:

1. Aho, Lam, Sethi & Ullman, *Compilers: Principles, Techniques, & Tools*, Pearson Education.

2. Kieth D. Cooper & Linda Troczon, *Engineering a Compiler*, Morgan Kaufmann.

3. Grune, Reeuwijk, Bal, Jacobs & Langendoen, *Modern Compiler Design*, Springer.

4. John Levine, *Flex & Bison*, O'Reilly.

BCAMJ604T : Cyber Security Theory: 40 Lectures

	Course Outcome
1.	To understands the conceptual and technical foundation cyber security.
2.	To exhibit knowledge to secure corrupted systems, protect personal data, and secure computer networks in an Organization
3.	To identify and analyze statutory, regulatory, constitutional, and

	organizational laws that affects the information technology professional.
4.	To apply case law and common law to current legal dilemmas in the technology field.
5.	To apply diverse viewpoints to ethical dilemmas in the information technology field and recommend appropriate actions,
6.	To understand principles of web security and to guarantee a secure network by monitoring and analyzing the nature of attacks through cyber/computer forensics software/tools.

1. Introduction (5 Lectures)
Security, Attacks, Computer Criminals, Security Services, Security Mechanisms.
2. Cryptography (20 Lectures)
Substitution ciphers, Transpositions Cipher, Confusion, diffusion, Symmetric, Asymmetric Encryption. DES Modes of DES, Uses of Encryption, Hash function, key exchange, Digital Signatures, Digital Certificates.
3. Program Security (4 Lectures)
Secure programs, Non malicious Program errors, Malicious codes, Virus, Trap doors, Salami attacks, Covert channels, Control against program
4. Threats. (4 Lectures)
Protection in OS: Memory and Address Protection, Access control, File Protection, User Authentication.
5. Security in Networks (4 Lectures)
Threats in Networks, Security Controls, firewalls, Intrusion detection systems, Secure e-mails
6. Administrating Security (3 Lectures)
Security Planning, Risk Analysis, Organizational Security Policy, Physical Security. Ethical issues in Security: Protecting Programs and data. Information and law.

Recommended Books:

1. C. P. Pfleeger, S. L. Pfleeger; *Security in Computing*, PHI
2. W. Stallings, *Network Security Essentials: Applications and Standards*, Pearson
3. Forouzan, Mukhopadhyay, *Cryptography And Network Security*, TMH
4. Stallings, *Cryptography and Network Security: Principles and Practice*, Pearson

BCAMJ604P : Cyber Security and Cyber Laws Lab Practical: 40 Lectures

	Course Outcome
1.	Design a simple cipher offering confidentiality
2.	Design a simple cipher that provides authentication and/or integrity and/or non-repudiation
3.	Demonstrate use of different network security tools
4.	Protect word document, worksheet etc from unauthorized users
5.	To apply diverse viewpoints to ethical dilemmas in the information technology field and recommend appropriate actions,
6.	To understand principles of web security and to guarantee a secure network by monitoring and analyzing the nature of attacks through cyber/computer forensics software/tools.

1. Demonstrate the use of Network tools: ping, ipconfig, ifconfig, tracert, arp, netstat, whois.
2. Use of Password cracking tools : John the Ripper, Ophcrack. Verify the strength of passwords using these tools.
3. Write a program to perform encryption and decryption of Caesar cipher.
4. Write a program to implement polyalphabetic cipher.
5. Write a program to perform encryption and decryption of a transposition cipher
6. Write a program to perform affine cipher.
7. Write a program to program to implement Rail fence cipher.
8. Write a program to implement one-time pad.
9. Write a program to implement RSA algorithm.
10. Write a program to implement RSA algorithm with big numbers.
11. Demonstrate sending of a protected word/worksheet document.
12. Demonstrate sending of a digitally signed document.

- END -