AMITY UNIVERSITY
UTTAR PRADESH

Course Title: Data Structure & Algorithm Design

L	T	P/S	SW/FW	TOTAL CREDIT UNITS
3		4		5

Course Level: PG
Course Code: CSE 601 **Credit Units:** 05

Course Objectives:
The objective of this course module is to describe the utilization of formal ADT representations, especially those on lists, sets, trees and graphs; time and space analysis of recursive and non-recursive algorithms, including graph and sorting algorithms; to learn techniques for designing algorithms using appropriate data structures, prove correctness and analyze their running times.

Pre-requisites:
Data Structures in C, Analysis and Design of Algorithm, and C Programming Language.

Course Contents/Syllabus:

	Weightage (%)
Module I :	
Overview of Data Structures: Stacks, Queues, linked lists, doubly linked lists, Applications. Analysis and Efficiency of algorithms, Asymptotic Notations, Time complexity of an algorithm, Analyzing Recursive Programs using various strategies. **Divide and Conquer Paradigm:** Divide and conquer recurrence equations and their solutions, Review of various sorting techniques and its time complexity.	22%
Module II :	
Trees: Basic terminology& its applications, Binary Trees, Binary Search Trees, Red-Black Trees, AVL Trees and B Trees, spanning trees: Prim's and Kruskal's algorithm. **Graphs:** Terminology, representations, traversals. **Basic graph algorithms:** Depth first search and Breadth first Search and its analysis, single source and all-pair shortest path problem.	22%
Module III:	22%
Comparison between Dynamic and Greedy Approach to solve a problem. **Dynamic Programming:** Matrix Chain Multiplication, Longest Common subsequence and 0/1 Knapsack Problem. **Greedy**	

Programming: Activity Selection Problem, Huffman Codes and fractional Knapsack Problem.	
Module IV :	19%
String Matching Algorithms: Naïve, Rabin Karp and Knuth Morris and Pratt paradigm.**Approximation Algorithms:** Vertex-Cover algorithm and Set Covering Problem.	
Module V :	15%
NP-Completeness: Polynomial Time, NP-completeness proofs and problems, NP-Hard and SAT problems.	

Student Learning Outcomes:
<u>Outcome 1</u>: Scientific &Computing foundation: ability to apply knowledge of mathematics, science, engineering and computing appropriate to the discipline.
<u>Outcome 2</u>: Analysis: ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
<u>Outcome 11</u>: Mathematical foundation: ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
<u>Outcome 12</u>: Software Construction: ability to apply design and development principles in the construction of software systems of varying complexity.

Pedagogy for Course Delivery:

1. Classroom teaching using White board and Presentations.
2. Assignments and Tutorials for continuous assessment.

Assessment/ Examination Scheme:

Theory L/T (%)	**Lab/Practical/Studio (%)**	**Total**
70	30	100

Theory Assessment (L&T):

Continuous Assessment/Internal Assessment					**End Term Examination**
Components (Drop down)	**Attendance**	**Class Test**	**Assignment**	**Presentation/Viva**	
Weightage (%)	5	10	7	8	70

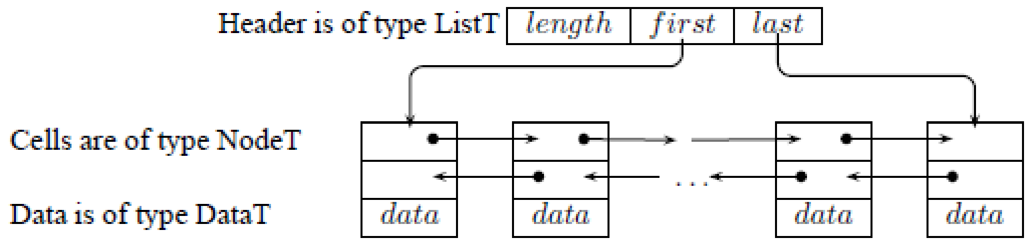
Lab

Continuous Assessment/Internal Assessment					End Term Examination
Components (Drop down)	**Attendance**	**Performance**	**Continuous Evaluation**	**Presentation/Viva**	
Weightage (%)	5	10	10	5	70

Course Lab

Based on Course Lab Credits, student is required to perform following assignments:

Assignment #	Assignment	
1	**Data Structure Concepts** 1. Write a function that receives marks scored by a student in 3 subjects and returns the average and percentage of these marks. Use call by value and call by reference concept to call this function from main () and print the results in main (). 2. Write a function to compute the distance between two points and use it to develop another function that will compute the area of the triangle whose vertices are A(x1,y1),B(x2,y2), and C(x3,y3).Use these functions to develop a function which returns a value 1 if the point (x,y) lies inside the triangle ABC, otherwise a value 0.	
2	**Stacks & Queues** 1 WAP to implement following operations on stack using link lists. a Insertion in Stack b Deletion from Stack c Display all elements of Stack 2 WAP to implement queue using Link Lists. a Enqueue in Queue b Dequeue in Queue c Display all elements of Queue	
3	**Stacks & Queues** 1. WAP to implement operations on queue using two stacks.	
4	**Link Lists** 1. WAP to implement Circular Link Lists	
5	**Doubly Link Lists** Read a paragraph containing words from an input file. Then create a doubly-linked list containing the distinct words read, and their occurrence frequency. This list should have the structure as	

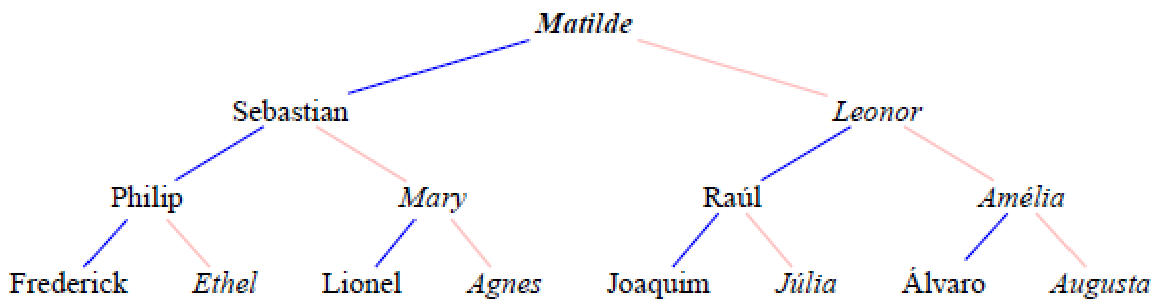
	depicted in below Figure. The data structures are defined as follows: **typedefstruct** { **int***length*; *NodeT* f irst*, **last* ; } *ListT*; **typedefstruct** { **char** **word*; **double** *frequency*; } *DataT*; **typedefstruct***node* { **struct***node* **prev*, **next* ; *DataT data*; } *NodeT*;  Header is of type ListT length first last Cells are of type NodeT Data is of type DataT data data . . . data data The list should be alphabetically ordered. Print the words and their frequencies in alphabetical ascending and descending order, case insensitive.	
6	**Sorting Techniques**	
	1. WAP to implement Quick sort 2. WAP to implement Heap sort	
7	**Sorting Techniques**	
	1. WAP to implement Counting Sort	
8	**Trees**	
	1. WAP to implement operations on Binary Trees using Link Lists 2. WAP to implement operations on Binary Search Trees using Link Lists	
9	**Open Ended Experiment**	
	Construct a family tree organized as follows: the root node holds a parents's name (names must be	

unique) as a key; each name is unique. Then, after reading pairs of names from a file of your choice, output the relations between pairs or persons.

Input. (<parentName>(<childName><childName>... <childName>))
...
?< name1 >, < name2 >
The angle brackets ('<', '>') do not exist in the input, they just mark changeable amounts. All names (i.e. <childName>,<parentName>,< name1 >,< name2 >are alphabetic strings with no spaces. '(', ')' are delimiters. '?' signals query about persons. Output is relation degree (i.e, child, parent, cousin, etc.). For the example tree in Figurebelow, a part of the input would be:

(Matilde (Sebastian Leonor))
(Sebastian (Philip Mary))
(Philip (Frederick Ethel))
(Raul (Joaquim Julia))
...
? Matilde, Agnes
? Agnes, Matilde
? Joaquim Julia

and part of output would be:
Matilde is grand-grand-mother/father for Agnes
Agnes is grand-grand-son/daughter for Matilde
Joaquim is sibling for Julia



10	**Trees**	
	1. WAP to implement operations on RB-Trees	
11	**Trees**	
	1. WAP to implement operations on AVL Trees	
12	**Trees**	
	1. WAP to implement operations on B+ Trees.	

13	**Graphs** 1. Implement an algorithm to find the *longest* cost simple path in a *directed* graph.	
14	**Graphs** 1. WAP to implement All Pair Shortest Path Algorithm	
15	**Graphs: Map Coloring** A map of an area of the world contains n countries. Each country is a neighbor of some other country. You are given m colors to color this map. Find all the possible country colorings using all the m colors, such a way that any two neighboring countries are colored differently.	
16	**Dynamic Programming** 1. WAP to find Least Common Sub-Sequence	
17	**Dynamic Programming** The input to the problem is a pair of strings of letters A = a1 . . . a_m and B = b1 . . . b_n. The goal is to convert A into B as cheaply as possible. The rules are as follows: • For a cost of 3 you can delete any letter. • For a cost of 4 you can insert a letter in any position. • For a cost of 5 you can replace any letter by any other letter.	
18	**Greedy Approach** 1. WAP to implement Activity Selection Problem.	

19	**String Matching Algorithm**
	1 Write a program to implement String matching algorithm on following data:
	T = xyxxyxyxyyxyxyxyyxyxyxxy
	P = xyxyyxyxyxx	

20	**Open Ended Experiment: Josephus Problem** In a certain counting-out game there are people standing in a circle waiting to be executed. After the first person is executed, certain numbers of people are skipped and one person is executed. Then again, people are skipped and a person is executed. The elimination proceeds around the circle (which is becoming smaller and smaller as the executed people are removed), until only the last person remains, who is given freedom. Implement the above game by taking names of people to be executed and arrange them in circle and find out the name of the person who is given freedom. Take the position to start and number of persons to skip as input from user.	

Text & References:

Text:

- Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, “Introduction to Algorithms”, MIT press and McGraw Hill, 2001.

- UdiManber, “Introduction to Algorithms: A Creative Approach”, Addison Wesley, 1989.
- Ellis Horowitz, Sartajsahni, “Fundamentals of Data Structures” Galgotia book source, New Delhi, 1983.
-

References:

- Ellis Horowitz, Sartajsahni, “Fundamentals of Algorithms” Galgotia book source, New Delhi, 1986.
- Jean paul Tremblay and paul G. Soresson, “An introduction to Data structures with applications” Mcgraw Hill International editions.
- Seymour Libschutxz, “Theory and problems of Data structures”, Mcgraw Hill International editions. (Schaum’s outline series).
- Aho, Hopcroft Ullman, “The design and analysis of computer algorithms” Addison Wesley publishing company
- Robert L Cruse : "Data Structures and Program Design" (Prentice Hall India ,3rd Edition 1999)
- ReemaThareja, “Data Structures using C”, Oxford Higher Education, New Delhi, 2011.