CS201T and CS201P: Data Structures Using C

Objective

The objective of the course is to present the students to learn how to create data structures in a computer language, such as C, to represent a collection of similar data, and how to process these data most efficiently for solving problems. After completion of this course, a student will be able to

- » understand and use the process of abstraction using a programming language such as 'C'
- » analyse step by step and develop algorithms to solve real world problems
- » implement various data structures viz. Stacks, Queues, Linked Lists, Trees and Graphs
- » understand various searching and sorting techniques and their processing efficiency.

It is expected that the student has basic knowledge of C language.

Outline of the Course

Minimum Class Hours			Exam time (Hours)		Marks				
					Theory		Practical		
Theory	Practical	Total	Theory	Practical	External	Internal	External	Internal	Total
60	40	100	2	3	45	15	30	10	100

Unit	Topic	Minimum Class Hours			Marks (Theory)
		Theory	Practical	Total	
I	Introduction to C	10	4	14	09
II	Linear Data Structures: Linked List, Stacks, Queues	15	9	24	09
III	Trees	15	9	24	09
IV	Graphs	10	9	19	09
V	Searching and Sorting	10	9	19	09
Total		**60**	**40**	**100**	**45**

Detailed Syllabus

Unit I: Introduction to C **10+4 Hours**

The C character set, Data types, Constants, Strings, Variables, Operators, Expressions, Statements Operators (arithmetic, unary, binary, ternary, relational, logical), Basic I/O Functions Decision making and looping: if ... else; switch; while, do ... while, for; nested loops; break; continue

Functions: Defining a Function, function prototypes, Call by value, Call by reference

Arrays: Declaration, initialisation, as arguments to functions, two-dimensional arrays

Pointers: Declaration, initialisation, address operator, indirection operator; Pointers and functions (taking pointers as arguments and returning pointers); Pointers and arrays (one-dimensional and two-dimensional)

Structures: Declaration, instantiation; member access operators (. and ->), functions and structures; Nested Structures; Array of Structures

Unit II: Linear Data Structures: Linked List, Stacks and Queues **15+9 Hours**

Data Type, Abstract Data Type, Data Structure, Complexity measures in terms of time and space; Big O notation. Linked List as a data structure (characteristics, advantages, disadvantages); operations on lists (creation, insertion, deletion, traversal, merging, splitting); doubly linked list, circular list; use of linked lists for polynomial representation and manipulation (addition and multiplication).

Stacks and Queues as data structures; implementation of stacks and queues using arrays and linked lists; Definitions of Circular Queue, Priority Queue; Application of stacks : Conversion of infix(containing arithmetic operators including exponential operator, and parenthesis) to postfix, evaluation of postfix expression.

Unit III: Trees **15+9 Hours**

Definition of tree as a data structure (Binary Trees and General Trees), Basic Terms (father, son, descendant, ancestor, height, depth, leaf, node, forest, ordered trees, strictly binary tree, complete binary tree, internal nodes, external nodes); Representation of trees using linked lists, Binary tree traversal methods (pre-order, in-order, post-order), recursive algorithms for traversal methods, Binary search trees (creation, insertion and deletion of a node), Height balanced (AVL) binary trees (construct and traverse an AVL tree)

Unit IV: Graphs **10+9 Hours**

Definition of a graph, Basic Terms (vertex, arc, directed, undirected, cardinality, finite and infinite graph, incidence, adjacency, indegree, outdegree, path length, weighted graph, connected graph, cyclic and acyclic graph, symmetric graph, complete graph, sub-graph); Graph representation : Adjacency matrix, adjacency lists, incidence matrix, adjacency multi-lists; Traversal schemes : Depth first search, Breadth first search (Recursive and non-recursive algorithms); Shortest Path algorithms (Dijkstra's), Spanning tree, Minimal spanning tree algorithms (Kruskal's algorithm)

Unit V: Searching and Sorting **10+9 Hours**

Linear and binary search and their complexity analysis; Hashing, Hash Functions (division method, mid square method, folding), Analysis of ideal hash function; Conflict resolution (linear and quadratic probe, double hashing, separate chaining); Analysis of collision resolution techniques; Sorting algorithms(Insertion, Selection, Bubble, Quick) and comparison of their time complexity.

Instructions for Paper Setter

The question papers will be set according to the following scheme

UNIT	Theory			Practical		
	To be set	To be answered	Marks	To be set	To be answered	Marks
I	2	1	09			
II	2	1	09			
III	2	1	09	2	1	10
IV	2	1	09	2	1	10
V	2	1	09	2	1	10
Total	10	5	45	6	3	30

Distribution of marks for practical

10% : Syntax and input/output screens
30% : Logic and efficiency (source code, pseudocode, and algorithm)
20% : Error trapping (illegal or invalid input, stack overflow, underflow, insufficient physical memory etc.)
20% : Completion
20% : Result

Recommended Books

Text Book

1. **S. Chattopadhyay, D. Ghosh Dastidar, M Chattopdhyay,** *Data Structures Through C Language,* BPB Publications, 2001
2. **Yashavant Kanetkar,** *Let us C* , BPB Publication

Reference

1. **Y. Langsam, M.J. Augenstein, A.M. Tenenbaum,** *Data Structures Using C and C++,* Second Edition, Prentice Hall of India, 2000
2. **Seymour Lipschutz,** *Theory and Problems of Data Sturctures,* Schaum's Outline Series, International Edition, MacGraw Hill, 1986
3. **Y.P. Kanetkar,** *Data Structures Through C Language,* BPB Publications, 2002

Practical Assignments

(Questions need not be restricted to this list)

1. Write a program to sum the following series:
   a) The first n natural numbers
   b) The first n odd natural numbers
   c) The first n even natural numbers
2. Write a program to sum the series : 2 * 3 – 3 * 5 + 4 * 7 + to n terms
3. Given a number, write a program using while loop to reverse the digits of the number. For example, the number 12345 should be written as 54321. (**Hint:** Use modulus operator to extract the last digit and the integer division by 10 to get the n-1 digit number from the n digit number.)
4. Write a program for sorting the elements of an array by using Selection sort, Bubble sort, Insertion sort.
5. Write a program to generate positive prime numbers.
6. Write a program to find the sum of row, column, and diagonals of the given matrix.
7. Write a program to find the largest number of the given matrix using function.
8. Write a program to sort all the elements of a matrix using function.
9. Write a menu-driven program to
   a) construct a singly linked list. Assume the information part of each node consists of only an integer key. Get input for each key from the keyboard. Assume the input is over when the user enters –1
   b) print the information from each node
   c) delete all nodes containing a given number
   d) Exit

6. Consider that L, a linked list of n integers, is given to you. Suppose, the nodes of the list are numbered from 1 to n. It is required to split the list L into 4 lists so that the first list contains the nodes of L numbered 1, 5, 9, 13 ... The second list contains the nodes numbered 2, 6, 10, 14 ... The third list contains the nodes of L numbered 3, 7, 11, 15, ... The fourth list contains the nodes of L numbered 4, 8, 12, 16 ... Write a program to create the list and perform the splitting.

7. Write a C function to insert a node appropriately to an already sorted list so that after insertion, the new list also becomes sorted. Take care of special cases such as inserting into an empty list. Use this function to write a program which accepts integers at the input and at the end produces a sorted list. Assume that if the integer read at the input is '0' then your program should stop.

8. Write a program to implement polynomial multiplication. Test your program by inputting the following two polynomials given below.

   10 P^8 + 14 P^6 - 8 P^5 - 3 P^4 + P^2

   3 P^4 + 5 P^3 - 2 P + 9

   (^ is to be read as "raised to")

   Store each term of the polynomial in a linked list in descending order of the index. Use separate linked lists for each polynomial. Obtain and store the product in a third linked list, and then print out all the three polynomials in a format similar to the one shown above, in descending order of index.

9. A bi-directional list is a list of elements that are linked in both ways. Both links are originating from a header. Construct a module with procedures for searching, inserting and deleting elements.

10. Write a program to represent a sparse matrix using linked list. Add together two such matrices, and display the original and resulting matrices in matrix form.

11. Write a menu-driven program to implement a stack ***using arrays***. The menu should have the following options:

    a) Push on to the stack
    b) Pop from the stack and print the value popped from the stack
    c) Merely print the value on top of the stack
    d) Exit

    Error trapping should be done for underflow and overflow. Available array space should be efficiently used (i.e. there cannot be overflow if there is more than 1 empty element in the array). Assume that the information part of a stack element is only an integer.

12. Write INSERT and DELETE functions in C language simulating insertion and deletion in circular queue which stores an array of characters.

13. A double-ended queue is a linear list in which additions and deletions may be made at either end of the queue. Write a C function to implement deque with desired functionality. Illustrate use of your function in an example problem, say, a queue of integers.

14. Devise a scheme to traverse a singly linked list in both directions by reversing the links during left to right traversal. Write a C program to implement this traversal scheme.

15. Write a C program to convert an expression from its infix form to its equivalent (a)postfix form, (b)prefix form. Assume the infix expression contains only operators +, -, /, *, ^. The operator '^' stands for exponentiation. The operands are all single digit integers. Display the resulting (a)postfix expression (b)prefix expression.

16. Write a program to input a postfix expression that consists of only single digit positive operands and the binary operators +, -, *, and /. Using a function, evaluate this postfix expression. The function should report if the postfix expression is invalid, else return its value. [For example, 242/-46*+7+ is a valid postfix expression (being the equivalent of the infix expression, 2-4/2+4*6+7) and its value is 31.00.]
17. Write a program to construct a binary search tree of integers using linked list. Assume the information part of each node consists of only an integer key. Get input for each key from the keyboard. Assume the input is over when the user enters –1. Next, print out the keys in ascending order of magnitude, using a non-recursive function.
18. Write a program to create a binary tree and to traverse the tree in
    a) pre-order
    b) in-order
    c) post-order
19. Implement a procedure for deleting an element *X* from a binary search tree.
20. Write a program to reconstruct a binary search tree given its pre-order and in-order traversal sequence.
21. Write a program to find the biggest and smallest item in a binary search tree.
22. Design and implement an algorithm for insertion of an element in AVL tree taking into account all possible conditions.
23. Represent a graph using adjacency matrix. Write a C procedure to transform an adjacency matrix based representation to a linked-list based representation.
24. Design a suitable representation so that a graph can be stored on a hard disk. Write a procedure adjacency matrix based representation.
25. Write a program to represent a graph and perform a non-recursive depth first search of an item in it.
26. Write a program to represent a graph and compute the shortest distance between two nodes in it.
27. Write a program to input some numbers into an array, and then sort them using various sorting techniques (selection sort, bubble sort, quick sort, radix sort) and compare their time-complexities.
28. Write a program to input some numbers (at least 128 numbers, the more the better) from a file into two arrays A and B. Sort array B. Perform the linear search in array A and binary search in array B for a given number. Repeat these as many times as user decides and compare the time-complexity of the two search methods on the average.
29. Design and implement an algorithm to delete an identifier *X* from a hash table which uses hash function *f* and linear open addressing to resolve collisions. Your deletion scheme must ensure that correct search is possible even after deletion.