Data Structures and Algorithms - I

Teaching Scheme	**Examination Scheme**
Lectures: 2 Hrs / Week	Assignment/Quizzes : 40 marks
	End Semester Exam : 60 marks

Course Outcomes

Students will be able to:

1. Write neat code by following coding standards, by selecting appropriate data structure and demonstrate a working solution for a given problem.
2. Think of all possible inputs to an application and handle all possible errors properly.
3. Analyze different possible solutions to a program and select the most efficient one.
4. Write an application requiring an effort of at least 500 lines of code to demonstrate a good working solution.
5. Demonstrate the ability to write reusable code and abstract data types.

Course Contents

Introduction to imperative programming. Syntax of a language for imperative programming. Input-output statements, data, data types, variables, constants, limitations of data types, type conversion, operators, control statements,compilation and execution as independent steps. Functions, Macros, Preprocessor directives, global, static, local variables, arrays and structures, pointers, pointer arithmetic, Dangling pointers and garbage memory, user defined types. writing code in multiple files, Coding standards

[6 Hrs]

Introduction to Data representation and files: Text and binary files, use of various libraries for handling files. Implementation of utilities like 'cat', 'cp', 'mv', etc

[3 Hrs]

Lists: Concept of Abstract Data types (ADT), List as ADT,Features of a Linear data structure, Dynamic memory allocation. self-referential structures, Concept of linked organization of data. Various implementations of List ADT as singly linked list, doubly linked list, circular linked list. Operations on linked lists: insert, delete, traverse, search etc. Applications of linked list: Representation & manipulations of polynomials/sets using linked list concept.

[5 Hrs]

Stacks and Queues: Stack and queue as ADT. Operations on stack and queue. Implementations using arrays and dynamic memory allocation. Application of stack for expression evaluation, expression conversion. Implementation of stack using queue and vice-versa. Recursion and stacks. Problems like maze and knight's tour.

[6Hrs]

Time Complexity Analysis. Characteristics of an algorithm. Analyzing programs. Frequency count. Time and space complexity. Big 'O' Ω, Θ, notation. Best, average and worst cases.

[2 Hrs]

Searching, Sorting Algorithms: Searching: linear and binary search algorithm. Searching using key-value in a sequence of records. Insertion, bubble, selection sort algorithms. Sort algorithms on a sequence of records using specified key.Comparative analysis of various searching and sorting algorithms.

[4 Hrs]

Applications: Design of data structures, modules, functions and algorithms for applications like a general-purpose random precision calculator like "bc", 'sort' utility with external sort, 'diff', 'grep', 'head', 'tail', 'dd', 'uniq', 'wc' etc.

[4 Hrs]

Text Books

- "Fundamentals of Data Structures in C", E. Horowitz, S. Sahni, S. Anderson-freed, Second Edition, 2008, University Press, ISBN 978-81-7371-605-8
- "The C Programming Language", B. Kernighan, D. Ritchie, Second Edition, 2015, Pearson Education India; ISBN 81-203-0596-5

Reference Books:

- "Data Structures using C", Y. Langsam, M. Augenstin and A. Tannenbaum, First Edition, 2002, Pearson Education Asia, ISBN 978-81-317-0229-1
- "C++: The Complete Reference", Herbert Schildt, Fourth Edition, 2002, The McGraw-Hill company, ISBN 0-07-222680-3
- "Fundamentals of Data Structures in C++", Ellis Horowitz, S. Sahni, D. Mehta, 2nd Edition, 2008, University Press, ISBN-10: 8173716064
- "An introduction to data structures with Applications", Jean-Paul Tremblay, Paul. G. Soresan, 2nd Edition, 1984, Tata Mc-Graw Hill International Editions, ISBN-0-07-462471-7

Data Structures and Algorithms –I Laboratory

Teaching Scheme
Laboratory: 2 Hrs / Week

Examination Scheme
Continuous evaluation: 50 Marks
Mini Project: 25 marks
End Semester Exam: 25 Marks

Course Outcomes

Students will be able to:

1. Write neat code by following coding standards, by selecting appropriate data structure and demonstrate a working solution for a given problem.
2. Think of all possible inputs to an application and handle all possible errors properly.
3. Analyze different possible solutions to a program and select the most efficient one.
4. Write an application requiring an effort of at least 500 lines of code to demonstrate a good working solution.
5. Demonstrate the ability to write reusable code and abstract data types.

Suggested List of Assignments

1. *Assignments for practice of language syntax, data types and programming constructs*
   a. Write a program to print a histogram of the frequencies of different characters in its input.
   b. Write a program to print sum of digits of a number accepted from user. Example: if the input number is 7654, output should be 7 + 6 + 5 + 4 = 22.
   c. Write a program to determine a string is a palindrome or not. A palindrome is a word, number, phrase, or other sequence of characters which reads the same backward as forward, such as madam, racecar
   d. Write a program in C to define a structure for a Customer bank account that holds information like Account Number, Name of account holder, balance, internet banking facility availed(Yes or No), Pin code (422001 to 422013), Account type(saving, recurring, deposit).
      i. Read account details for N customers
      ii. Classify the customer as Golden, Silver and General based on the following criteria: Golden customers: Balance> 10, 00000, Silver Customers: Balance >500000 and <500000, General customers: Balance <500000
      iii. Display the list of customers availing the Internet banking facility.
      iv. Display the customers belonging to a particular geographical location depending on postal code.
      v. Display the customer list as per their account type
   e. Write a function squeeze (s, c), which removes all occurrences of the character c from the string s.
   f. Write a C program to accept a string and change the case of each character of the string. Example " THIs Is a C Program" changes to "thiSiS A c pROGRAM"
   g. Write a C program to swap two numbers using a function.
2. Draw a diagram of data structures created by *given code* using a tool like xfig.
3. Write a program to compute x^y based on using base-3 presentation of a number. In the program, write a function which computes x^y

4. Write a program to remove duplicate doubles from an array of doubles. In the program, write a function which accepts an array of doubles and removes the duplicates from the array and has return type void.
5. Compare the time complexity of two sorting algorithms, by following the given steps. Create a set of data files with count of integers varying into thousands and millions. Sort the files using both the algorithms. Plot graph of the time taken by both the programs using tool like *gnuplot*. Compare the graphs and comment on the time complexity theoretically predicted and practically observed.
6. Write a function which evaluates an infix expression, without converting it to postfix. The input string can have spaces, (,) and precedence of operators should be handled.
7. Implement a queue (that is write queue.c and queue.h only) of characters, such that on an enqueue, the char is added at the end of queue, and on a dequeue the first element is taken out, but the queue uses only a 'head' pointer and not a 'tail pointer.
8. Write an data type called "Integer". The data type should represent integers of unlimited length.
9. Implement a data type to represent a Polynomial with the operations like create an empty polynomial, insert an entry into polynomial, add two polynomials and return the result as a polynomial, evaluate a polynomial, etc.
10. Implement a Set data type using sequentially linked structures. Support operations like create an empty set, insert an element into set, do a union of two sets and return results as a set, etc.
11. Write a sorting program with the following features: Reads data from a text file and sorts it alphabetically by default. If the file has data in rows and columns (separated by space or tab) then allows sorting on a particular column. Allows any sort using numeric or alphbetical ordering.
12. **Mini-project**: Write an application demonstrating your skills in defining a problem, writing down the requirements carefully, designing a modular solution with clear separation of abstract data types and their use, design of proper function prototypes and division of work among functions. The application can be a unix command re-implemented e.g grep, sort, diff, calculator etc. or a task defined after discussion with the instructor.