2022-23

جب کوئی قوم فن اور علم سے عاری ہو جاتی ہے تو وہ غربت کو دعوت دیتی ہے اور جب غربت آتی ہے تو وہ ہزاروں جرائم کو جنم دیتی ہے۔

"When a nation becomes devoid of art and learning, it invites poverty and when poverty comes it brings in its wake thousands of crimes."
-Sir Syed Ahmad Khan

BCA II Semester Lab Manual

CABSMJP02

Lab-II

DEPARTMENT OF COMPUTER SCIENCE

ALIGARH MUSLIM UNIVERSITY, ALIGARH

2022-2023

Credits

The following lab manual up-gradation committee updated the lab manual:

- ❒ *Prof. Aasim Zafar (Chairperson)*
- ❒ *Dr. Suhel Mustajab*
- ❒ *Dr. Arman Rasool Faridi*
- ❒ *Dr. Faisal Anwar*
- ❒ *Dr. Mohammad Nadeem*

The following committee member originally design the lab manual:

- ❒ *Prof. Mohammad Ubaidullah Bokhari*
- ❒ *Dr. Arman Rasool Faridi*
- ❒ *Dr. Faisal Anwer*
- ❒ *Prof. Aasim Zafar (Converner)*

Design & Compilation:

- ❒ *Dr. Shakeel Ahamad*

Revised Edition: ***January, 2023***

Department of Computer Science,
A.M.U., Aligarh (U.P.) India

COURSE TITLE: Lab -II
COURSE CODE: CABSMJP02
CREDIT: 02
PERIODS PER WEEK:03
CONTINUOUS ASSESSMENT:60
EXAMS: 40

COURSE DESCRIPTION

This manual is intended for Second-year students in Data structure through C. This manual typically contains practical/Lab Sessions related to Data structure through C, covering various aspects to enhance subject understanding. In computer science, a data structure is a way of organizing and storing data in a computer program so that it can be accessed and used efficiently. Data structures manage large amounts of data, enabling efficient searching, sorting, insertion, and deletion.

Data structures can be categorized into two types: primitive data structures and non-primitive data structures. Primitive data structures, such as integers, floating-point numbers, characters, and Booleans, are the most basic in a programming language. Non-primitive data structures are complex data structures built using primitive data types, such as arrays, linked lists, stacks, queues, trees, graphs, and hash tables.

The choice of data structure for a particular task depends on the type and amount of data to be processed, the operations that need to be performed on the data, and the efficiency requirements of the program. Efficient use of data structures can greatly improve the performance of a program, making it faster and more memory-efficient. A data structure is a particular way of organizing data in a computer so that it can be used effectively. The idea is to reduce the space and time complexities of different tasks

The lab manual will provide step-by-step instructions on completing the exercises, sample code, and expected output. By the end of the course, students

should have a solid understanding of the principles of data structures and algorithms and be able to apply this knowledge to solve real-world problems using the C programming language.

Students are advised to thoroughly review this manual rather than only the topic mentioned in the syllabus, as practical aspects are the key to understanding and conceptualising theoretical aspects covered in the books.

CONTENT

This course is designed to allow the students to learn Data structure through c. This course is intended to develop a deep understanding of various operations on data structure such as searching, sorting, insertion, deletion and traversing and provides students the idea of a new programming language. Apart from basic concepts, it helps to understand the concept of lists, tuples, dictionaries and file handling.

OBJECTIVES

This course is designed to help students in:

- ❒ Learning basic concepts of Data structures.
- ❒ Developing data structures using c programming.

OUTCOMES

After completing this course, the students would be able to:

- ❒ Understand and implement the different data structures concepts of Using c programming.
- ❒ Understand and implement basic operations using C programming.

- Write, debug and execute programs to solve various data structures problems.

RULES AND REGULATIONS

Students are required to adhere to the following rules strictly.

- The students must complete the weekly activities/assignments well in time (i.e., within the same week).
- The students must maintain the Lab File of their completed activities/assignments in the prescribed format (**Appendix-1**).
- The students must get the completed weekly activities/assignments checked and signed by the concerned teachers in the Lab in the immediate succeeding week. Failing which activities/assignments for that week will be treated as incomplete.
- At least **TEN (10)** such timely completed and duly signed weekly activities/assignments are compulsory, failing which students will not be allowed to appear in the final Lab Examination.
- The students need to submit the following deliverables for each exercise duly signed by the Teacher:
  - Coding
  - Input /Output
- Each question will be evaluated on a scale of 10 points, 7 for Coding and 3 for Input/Output.
- The students must ensure that each question is assessed and signed by the Teacher in the week/time.
- Late submissions would not be accepted after the due date.
- Cooperate, collaborate and explore for the best individual learning outcomes, but copying is strictly prohibited.

APPENDIX-1

Template for the Index of Lab File

WEEK NO		PROBLEMS WITH DESCRIPTION	PAGE NO.	SIGNATURE OF THE TEACHER WITH THE DATE
1	1#			
	2#			
	3#			
2	1#			
	2#			
	3#			
3	1#			
	2#			
	3#			
4	1#			
	2#			
	3#			

Note: *The students should use Header and Footer mentioning their roll no. & name in footer and page no in header.*

WEEK #1

OBJECTIVES

- *To learn the concept of an array and string in C.*

OUTCOMES

After completing this, the students would be able:

- *To explain the basic features of the Array to manipulate the string.*

PROBLEMS

1# Write a C Program to Remove All Vowels from a String.

2# Write a C Program to Remove Given Word from a String

3# Write a C to find the Number of Vowels, Consonants, Digits and White Space Character

WEEK #2

OBJECTIVES

- *To learn the basic concept of Structures in c programming*

OUTCOMES

After completing this, the students would be able:

- *To explain the basic features of the Structures.*

PROBLEMS

1# Write a C Program to Store Students' Information on Using a Structure

2# Write a C to add Two Distances (in Inch-Feet) System Using Structures

3# Write a C to add Two Complex Numbers by Passing Structure with Function

4# Write a C to Calculate the Difference Between the Two Time Periods (HH: MM: SS)

5# Write a C Program to Store Information Using Structures, Pointer with Dynamically Memory Allocation.

.

WEEK #3

OBJECTIVE

- *To learn the complex problem of the Array.*

OUTCOMES

After completing this, the students would be able:

- *To learn how to perform various operations on the Array.*

.

PROBLEMS

Write a C program that uses functions to perform the following operations on an array

1# Create an array of size 25 and insert the element of your choice

2# Delete the element at the beginning, middle, and end and adjust the Array.

3# Insert the element at the particular locations in the Array.

4# Display all the elements

WEEK #4

OBJECTIVES

- *To learn the concept of various functions on a linked list.*

OUTCOMES

After completing this, the students would be able:

- *To explain the linked List's concept creation, deletion, and insertion.*

PROBLEMS

Write a C program that uses functions to perform the following operations on a singly linked list

1# Creation

2# Insertion

   a. At beginning
   b. At middle
   c. At end

3# Deletion

   a. At beginning
   b. At middle
   c. At end

4# Traversal.

WEEK #5

OBJECTIVES

- *To learn the concept of various functions on a double-linked list.*

OUTCOMES

After completing this, the students would be able:

- *To explain the concept creation, deletion, and insertion of the linked List*

PROBLEMS

Write a C program that uses functions to perform the following operations on a doubly linked list

1# Creation

2# Insertion

   a. At beginning
   b. At middle
   c. At end

3# Deletion

   a. At beginning
   b. At middle
   c. At end

4# Traversal.

WEEK #6

OBJECTIVES

- *To learn the concept of various functions on a circular linked List.*

OUTCOMES

After completing this, the students would be able:

- *To explain the concept creation, deletion, and insertion of the circular linked List*

PROBLEMS

Write a C program that uses functions to perform the following operations on a circular linked List

1# Creation

2# Insertion

3# Deletion

4# Traversal.

WEEK #7

OBJECTIVES

- *To learn the implementation of the stack using the Array and linked lists*

OUTCOMES

After completing this, the students would be able to:

- *To explain the concept creation, Push, and Pop operation on the stack*
- *Application of the stack in the expression conversion.*

PROBLEMS

1# Write a C program that implements stack (its operations) using

a. Arrays

b. Linked List (Pointers).

2# Write a C program to convert the expression Infix to Postfix.

3# Write a C program to evaluate postfix expression.

4# Write a C Convert the following infix expression into postfix form.

(A + B) * (C + B) * (E l F).

WEEK #8

OBJECTIVES

- *To learn the implementation of the Queue using the Array and linked lists*

OUTCOMES

After completing this, the students would be able to:

- *To explain the concept creation, insertion, and deletion operation on the stack*

PROBLEMS

1# Write a C program that implements Queue (its operations) using

a. Arrays

b. Linked List (Pointers).

WEEK #9

OBJECTIVES

- *To learn about circular Queues.*

.

OUTCOMES

After completing this, the students would be able to:

- *To construct the circular Queue and its operations*
- *Various searching algorithms*

PROBLEMS

1# Write a C program that implements Circular Queue using arrays.

2# Write a C program that uses both recursive and non-recursive functions to perform the following searching operations for a key value in a given list of integers:

a. Linear search

b. Binary search.

WEEK #10

OBJECTIVES

- *To learn about the various sorting algorithms.*

.

OUTCOMES

After completing this, the students would be able:

- *To learn the concept of sorting and its mechanisms.*

PROBLEMS

1# Write a C program that implements the following sorting

a. Bubble sort

b. Selection sort

c. Quick sort.

WEEK #11

OBJECTIVES

- *To learn about the various sorting algorithms.*

.

OUTCOMES

After completing this, the students would be able:

- *To learn the concept of sorting and its mechanisms.*

PROBLEMS

Write a C program that implements the following

1# Insertion sort

2# Merge sort

3# Heap sort.

WEEK #12

OBJECTIVE

- *To learn the concept of the binary search tree.*

.

OUTCOME

After completing this, the students would be able:

- *To create, insert and delete in the binary search tree.*

PROBLEMS

Write a C program to perform the following operations:

1# Insert an element into a binary search tree.

2# Delete an element from a binary search tree.

3# Search for a key element in a binary search tree.

WEEK #13

OBJECTIVE

- *To learn the concept of the AVL tree.*

.

OUTCOME

After completing this, the students would be able:

- *To Create, insert and delete in the AVL tree.*

PROBLEMS

1# Write a C program to implement the tree traversal methods.

2# Write a C program to perform the following operations:

a. Insert an element into an AVL tree.

b. Delete an element from an AVL tree.

c. Search for a key element in an AVL tree.

WEEK #14

OBJECTIVE

- *To learn the linked list applications.*

.

OUTCOME

After completing this, the students would be able:

- *To learn the implementation of various functions of a dictionary using the ADT.*

PROBLEMS

Write a C program to implement all the functions of a dictionary (ADT) using Linked List.