Lab Manual

B.Tech, 3rd Semester

Data Structure Laboratory (CS 2091)

School of Computer Engineering

KIIT, Deemed to be University

Lab Manual Design Committee:

- Prof. Rajat Kumar Behera
- Prof. (Dr.) Alok Kumar Jagadev
- Dr. Amulya Ratna Swain
- Dr. Arup Abhinna Acharya
- Dr. Satarupa Mohanty

The committee thankfully acknowledges the efforts and contributions of the following faculties:

- Prof. Harish Kumar Patnaik
- Prof. Suchismita Das
- Dr. Jagannath Singh
- Dr. Jitendra Kumar Rout
- Prof. Mohit Ranjan Panda
- Prof. Sujoy Datta
- Prof. Shaswati Patra
- Prof. Arup Sarkar
- Prof. Divya Kumari
- Prof. Ronali Padhy
- Prof. Lipika Mohanty
- Prof. Lipika Dewangan
- Prof. Santosh Kumar Baliarsingh
- Prof. Jhalak Hota
- Prof. Sagar Patnaik
- Prof. Gananath Bhuyan

Compilation:

- Prof. Rajat Kumar Behera

Prof. Rajat Kumar Behera
DSA Lab Coordinator

Rules and Regulations

Students are required to strictly adhere to the following rules.

- The students must maintain the soft copy of lab file for the completed activities/assignments.
- The students must complete the weekly activities well in time (i.e., within the same week).
- The students must get the completed weekly activities/assignments checked by the concerned teachers in the immediate succeeding week. Failing which the activities for that week will be treated as incomplete.
- All assignments must be timely completed, failing which students will not be allowed to appear in the lab sessional examination.
- LA stands for lab assignment and HA stands for home assignment. LAs are expected to be completed during the lab hours and to be maintained in the lab record. However, HAs are for the practice for the betterment of the coding skills. They are not necessarily to be written in the lab record. If students completed all LAs during the lab hours, they should proceed to solve HAs in the current lab hours.
- All programs to be developed using C.
- The lab expects highest level of academic honesty from the students during the various activities listed below in evaluation scheme. Any sort of malpractice will be dealt with seriously and will be referred to disciplinary committee.
- It is important to understand the fine line between copying programs and discussing programming ideas. The latter is encouraged and you should maintain high academic honesty standards by mentioning that name of your fellow classmate with whom you discussed in the code, as a comment. However, the course expects that each student shows individually that they have met the learning outcomes of the course.
- Evaluation Scheme:

Sr#	**Area**	**Mark**	**#**	**Total**
1	**Internal Sending**			
1.1	Lab record evaluation	1	10	10
1.2	Quiz	5	2	10
1.3	Viva	1	10	10
1.4	Program Execution	2	10	20
1.5	Attendance	1	10	10
Total				60
2	**End Term**			
2.1	Program Execution	5	2	10
2.2	Viva	15	1	15
2.3	Quiz	15	1	15
Total				40

Data Structure Lab

Objectives:

The objective of this lab is to teach students various data structures and to explain them algorithms for performing various operations on these data structures. This lab complements the data structure and algorithm course (CS-2001). Students will gain practical knowledge by writing and executing programs in C using various data structures such as arrays, linked lists, stacks, queues, trees, graphs, searching and sorting.

Learning Outcomes:

Upon the completion of data structure and algorithm lab, the student will be able to:

- Write, execute and debug programs in C to solve problems.
- Choose and implement efficient data structures and apply them to solve problems.

Syllabus:

The assignments are from the following list:

- Array, pointer with Dynamic Memory Allocation.
- Structure, Linked List
- Stack
- Queue
- Trees
- Graphs
- Searching
- Sorting

Lab SOP: Amid COVID-19, the lab will be operated in online mode, hence students are advised to make necessary arrangement for uninterrupted internet connectivity. The program execution will be corrected (towards end) or verified (any point of time) by the technical assistant/faculty via screen sharing using video communication app. It's the responsibility of students to verify the programs to earn score. Lab attendance will be taken using video communication app during the lab hours (at any point of time), so students are advised to be online throughout the session. GCC compiler is to be used for compiling and running the C programs.

Prof. Rajat Kumar Behera
DSA Lab Faculty

Week #1

Objectives:

- To learn writing, executing and debugging programs using array.
- To learn writing, executing and debugging programs using pointers.
- To learn writing, executing and debugging programs using dynamic memory allocation.

Outcomes:

- After completing this, the students would be able to develop, compile, debug, and fix errors of C programs based on array, pointers and dynamic memory allocation.

LAs:

- WAP to find out the smallest and largest element stored in an array of n integers.
- WAP to reverse the contents of a dynamic array of n elements.
- WAP to search an element in a dynamic array of n numbers.
- WAP to sort a dynamic array of n numbers.
- Given an unsorted dynamic array of size n, WAP to find and display the number of elements between two elements a and b (both inclusive). E.g. Input : arr = [1, 2, 2, 7, 5, 4], a=2 and b=5, Output : 4 and the numbers are: 2, 2, 5, 4.
- Given a dynamic array, WAP to print the next greater element (NGE) for every element. The next greater element for an element x is the first greater element on the right side of x in array. Elements for which no greater element exist, consider next greater element as -1. E.g. For the input array [2, 5, 3, 9, 7], the next greater elements for each elements are as follows.

Element	NGE
2	5
5	9
3	9
9	-1
7	-1

- Let A be nXn square dynamic matrix. WAP by using appropriate user defined functions for the following:

  a) Find the number of nonzero elements in A
  b) Find the sum of the elements above the leading diagonal.
  c) Display the elements below the minor diagonal.
  d) Find the product of the diagonal elements.

HAs:

- Given an unsorted dynamic array arr and two numbers x and y, find the minimum distance between x and y in arr. The array might also contain duplicates. You may assume that both x and y are different and present in arr.
  Input: arr[] = {3, 5, 4, 2, 6, 5, 6, 6, 5, 4, 8, 3}, x = 3, y = 6
  Output: Minimum distance between 3 and 6 is 4.
- WAP to find out the second smallest and second largest element stored in a dynamic array.
- WAP to arrange the elements of a dynamic array such that all even numbers are followed by all odd numbers.
- Write a program to replace every element in the dynamic array with the next greatest element present in the same array.
- WAP to replace every dynamic array element by multiplication of previous and next of an n element.
- WAP to sort rows of a dynamic matrix having m rows and n columns in ascending and columns in descending order.
- WAP to find out the k^{th} smallest and k^{th} largest element stored in a dynamic array of n integers, where k<n.
- WAP to find the largest number and counts the occurrence of the largest number in a dynamic array of n integers using a single loop.
- You are given an array of 0^s and 1^s in random order. Segregate 0^s on left side and 1^s on right side of the array. Traverse array only once.
- WAP to swap all the elements in the 1st column with all the corresponding elements in the last column, and 2nd column with the second last column and 3rd with 3rd last etc. of a 2-D dynamic array. Display the matrix.
- WAP to arrange the elements of a dynamic array such that all even numbers are followed by all odd numbers using a single loop.

Week #2

Objectives:

- To learn writing, executing and debugging programs using structure.
- To learn writing, executing and debugging programs using single linked list.

Outcomes:

- After completing this, the students would be able to develop, compile, debug, and fix errors of C programs based on structure and linked list.

LAs:

- WAP to store n employee's data such as employee name, gender, designation, department, basic pay. Calculate the gross pay of each employees as follows:
  Gross pay = basic pay + HR + DA
  HR=25% of basic and DA=75% of basic.
- WAP to add two distances (in km-meter) by passing structure to a function.
- Write a menu driven program to perform the following operations in a single linked list by using suitable user defined functions for each case.
  a) Traversal of the list.
  b) Check if the list is empty.
  c) Insert a node at the certain position (at beginning/end/any position).
  d) Delete a node at the certain position (at beginning/end/any position).
  e) Delete a node for the given key.
  f) Count the total number of nodes.
  g) Search for an element in the linked list.
  Verify & validate each function from main method.
- WAP to display the contents of a linked list in reverse order.
- WAP to print m^{th} node from the last of a linked list of n nodes.

HAs:

- WAP to search an element in a simple linked list, if found delete that node and insert that node at beginning. Otherwise display an appropriate message.
- WAP to count the number of occurrences of an element in a linked list of n nodes.
- WAP to reverse the first m elements of a linked list of n nodes.
- WAP to remove duplicates from a linked list of n nodes.
- Given a linked list which is sorted, WAP to insert an element into the linked list in sorted way.
- WAP to find number of occurrences of all elements in a linked list.

- WAP to modify the linked list such that all even numbers appear before all the odd numbers in the modified linked list.
- WAP to check whether a singly linked list is a palindrome or not.
- A linked list is said to contain a cycle if any node is visited more than once while traversing the list. WAP to detect a cycle in a linked list.
- WAP to reverse only even position nodes in a singly linked list.
- WAP to swap kth node from beginning with kth node from end in a Linked List
- Given a linked list, write a function to reverse every k nodes. (where k is an input to the function). If a linked list is given as 12->23->45->89->15->67->28->98->NULL and k = 3 then output will be 45->23->12->67->15->89->98->28->NULL.
- Given a singly linked list, rotate the linked list counter-clockwise by k nodes. Where k is a given positive integer. For example, if the given linked list is 10->20->30->40->50->60 and k is 4, the list should be modified to 50->60->10->20->30->40. Assume that k is smaller than the count of nodes in linked list.
- WAP to remove the duplicates in a sorted double linked list.

Week #3

Objectives:

- To learn writing, executing and debugging programs related to sparse matrix.
- To learn writing, executing and debugging programs related to polynomial.

Outcomes:

- After completing this, the students would be able to develop, compile, debug, and fix errors of C programs based on sparse matrix and polynomial.

LAs:

- WAP to create a linked list that represents a polynomial expression with single variable (i.e. $5x^7-3x^5+x^2+9$) and display the polynomial by using user defined functions for creation and display.
- WAP by modifying the LA1 program to add two polynomials with single variable. Use the same function in LA1 written for creation & display operations and write a new function for addition operations.
- A matrix m × n that has relatively few non-zero entries is called sparse matrix. It may be represented in much less than m × n space. An m × n matrix with k non-zero entries is sparse if k << m × n. It may be faster to represent the matrix compactly as a list of the non-zero indexes and associated entries. WAP to represent a sparse matrix using linked list.
- WAP to find out the transpose of a sparse matrix.
- WAP to determine whether the given matrix is a sparse matrix or not.

HAs:

- WAP to determine whether the given matrix is a lower triangular or upper triangular or tri-diagonal matrix.
- WAP by modifying the LA1 program to multiply two polynomials with single variable. Use the same function in LA1 written for creation & display operations and write a new function for multiplication operations.
- WAP to add two sparse matrixes.
- WAP to multiply two sparse matrixes.
- WAP to create a linked list that represents a polynomial expression with double variables (e.g.: $4x^2y^3-3xy+x-5y+7$) and display the polynomial by using user defined functions for creation and display.

Week #4

Objectives:

- To learn writing, executing and debugging programs using double linked list.
- To learn writing, executing and debugging programs using circular linked list.

Outcomes:

- After completing this, the students would be able to develop, compile, debug, and fix errors of C programs based on double linked list and circular linked list.

LAs:

- WAP to create a double linked list of n nodes and display the linked list by using suitable user defined functions for create and display operations.
- WAP to reverse the sequence elements in a double linked list.
- Write a menu driven program to perform the following operations in a double linked list by using suitable user defined functions for each case.
  a) Traverse the list forward,
  b) Traverse the list backward,
  c) Check if the list is empty
  d) Insert a node at the certain position (at beginning/end/any position)
  e) Delete a node at the certain position (at beginning/end/any position)
  f) Delete a node for the given key, g) Count the total number of nodes,
  h) Search for an element in the linked list
  Verify & validate each function from main method
- WAP to create a single circular double linked list of n nodes and display the linked list by using suitable user defined functions for create and display operations.

HAs:

- WAP to create a double circular double linked list of n nodes and display the linked list by using suitable user defined functions for create and display operations.
- Write a menu driven program to perform the following operations in a single circular linked list by using suitable user defined functions for each case.
  a) Traverse the list
  b) Check if the list is empty
  c) Insert a node at the certain position
  d) Delete a node at the certain position
  e) Delete a node for the given key
  f) Count the total number of nodes
  g) Search for an element in the linked list

Verify & validate each function from main method.

- WAP to remove the duplicates in a sorted double linked list.
- WAP to convert a given singly linked list to a circular list.
- WAP to implement a doubly linked list by using singly linked.
- WAP to print the middle of a double linked list.
- Given a double linked list, rotate the linked list counter-clockwise by k nodes. Where k is a given positive integer. For example, if the given linked list is 10->20->30->40->50->60 and k is 4, the list should be modified to 50->60->10->20->30->40. Assume that k is smaller than the count of nodes in linked list.

Week #5

Objectives:

- To learn writing, executing and debugging programs related to stack using arrays.
- To learn writing, executing and debugging programs related to stack using linked list.

Outcomes:

- After completing this, the students would be able to develop, compile, debug, and fix errors of C programs based on stack.

LAs:

- WAP Write a menu driven program to perform the following operations of a stack using array by using suitable user defined functions for each case.
  a) Check if the stack is empty b) Display the contents of stack c) Push d) Pop
- WAP Write a menu driven program to perform the following operations of a stack using linked list by using suitable user defined functions for each case.
- WAP to convert an infix expression into its equivalent postfix notation
- WAP to convert an infix expression into its equivalent prefix notation.
- Two brackets are considered to be a matched pair if the an opening bracket (i.e., (, [, or {) occurs to the left of a closing bracket (i.e.,),], or }) of the exact same type. There are three types of matched pairs of brackets: [], {}, and (). A matching pair of brackets is not balanced if the set of brackets it encloses are not matched. WAP to determine whether the input sequence of brackets is balanced or not. If a string is balanced, it print YES on a new line; otherwise, print NO on a new line.
  Example: Input: {[()]} and Output: YES
  Input: {[(])} and Output: NO

HAs:

- WAP to reverse a stack with using extra stack.
- WAP to sort the elements inside a stack using only push and pop operation. Any number of additional stacks may be used.
- WAP using a function that sort an array of integers using stacks and also uses bubble sort
- In a party of N people, only one person is known to everyone. Such a person may be present in the party, if yes, (s)he doesn't know anyone in the party. We can only ask questions like "does A know B? ". WAP to find the stranger (celebrity) in minimum number of questions.
- WAP exhibiting Tower of Hanoi (recursive).

- WAP to implement a stack which will support three additional operations in addition to push and pop by modifying LA 1.

  a) peekLowestElement - return the lowest element in the stack without removing it from the stack

  b) peekHighestElement - return the highest element in the stack without removing it from the stack

  c) peekMiddleElement - return the (size/2+1)th lowest element in the stack without removing it from the stack.

Week #6

Objectives:

- To learn writing, executing and debugging programs related to queue using arrays.
- To learn writing, executing and debugging programs related to queue using linked list.
- To learn writing, executing and debugging programs using circular queue.
- To learn writing, executing and debugging programs using deques.

Outcomes:

- After completing this, the students would be able to develop, compile, debug, and fix errors of C programs based on queue.

LAs:

- Write a menu driven program to implement queue operations such as Enqueue, Dequeue, Peek, Display of elements, IsEmpty, IsFull using static array.
- Write a menu driven program to implement queue operations such as Enqueue, Dequeue, Peek, Display of elements, IsEmpty using linked list.
- Write a menu driven program to implement circular queue operations such as Enqueue, Dequeue, Peek, Display of elements, IsEmpty, IsFull using static array.
- Write a menu driven program to implement Deques (both Input-restricted and Output-restricted) operations such as Enqueue, Dequeue, Peek, Display of elements, IsEmpty, IsFull using static array.
- Write a menu driven program to implement circular queue operations such as Enqueue, Dequeue, Peek, Display of elements, IsEmpty using linked list.
- Write a menu driven program to implement circular queue operations such as Enqueue, Dequeue, Peek, Display of elements, IsEmpty using linked list.

HAs:

- A stack data structure is given with push and pop operations. WAP to implement a queue using instances of stack data structure and operations on them.
- A queue data structure is given with enqueue and dequeue operations. WAP to implement a stack using instances of queue data structure and operations on them.
- Write a menu driven program to implement Deques (both Input-restricted and Output-restricted) operations such as Enqueue, Dequeue, Peek, Display of elements, IsEmpty using linked list.
- WAP using a function to reverse a queue by using stack.
- Given one queue data structure. WAP to implement stack using only one queue data structure.

Week #7

Objectives:

- To learn writing, executing and debugging programs related to linear search.
- To learn writing, executing and debugging programs related to binary search.
- To learn writing, executing and debugging programs related to hashing.

Outcomes:

- After completing this, the students would be able to develop, compile, debug, and fix errors of C programs based on searching.

LAs:

- WAP to read an array of integers and search for an element using linear search.
- WAP to read an array of integers and search for an element using binary search.
- Given an array container and integer hunt. WAP to find whether hunt is present in container or not. If present, then triple the value of hunt and search again. Repeat these steps until hunt is not found. Finally return the value of hunt.
  Input: container = {1, 2, 3} and hunt = 1 then Output: 9
  Explanation: Start with hunt = 1. Since it is present in array, it becomes 3. Now 3 is present in array and hence hunt becomes 9. Since 9 is a not present, program return 9.
- WAP illustrating chain hashing (Separate chaining with linked list).
- WAP illustrating following collision resolution technique:
  - "Open Addressing – Linear Search"
  - "Open Addressing - Quadratic Probing"
  - "Open Addressing - Double Hashing".
- Given a sorted array of length n, WAP to find the number in array that appears more than or equal to n/2 times. It can be assumed that such element always exists.
  Input: 2 3 3 4 Output: 3
  Input: 3 4 5 5 5 Output: 5

HAs:

- WARP (Write a Recursive Program) to search an element in a dynamic array of n integers using linear search.
- WARP using recursion to search an element in a dynamic array of n integers using binary search.
- WAP illustrating the usage of "Hash Functions" namely Folding, Mid-square, Division, Subtraction, Digit extraction and Rotation Hashing methods.

Week #8

Objectives:

- To learn writing, executing and debugging programs related to sorting.

Outcomes:

- After completing this, the students would be able to develop, compile, debug, and fix errors of C programs based on sorting.

LAs:

- WAP to sort an array of n dates in an ascending order using Bubble sort. Date structure is {day, month, year }
- WAP to sort an array of n floats in an ascending order using selection sort.
- WAP to sort an array of n integers in a descending order using insertion sort.
- WAP to sort an array of n integers in an ascending order using merge sort.
- WAP to sort an array of n doubles in a descending order using quick sort.

HAs:

- WAP sort the n names in an alphabetical order.
- WAP demonstrating bubble sort using linked list.
- WAP to find the maximum difference between any two elements.
- WAP to sort an array of n integers in an ascending order using radix sort.
- WAP to sort an array of n integers in an ascending order using Heap sort.
- WAP to input two arrays X and Y of positive integers, find number of pairs such that x^y > y^x where x is an element from X and y is an element from Y.

  Examples:
  - Input: X = {2, 1, 6}, Y = {1, 5}, Output: 3, There are total 3 pairs where pow(x, y) is greater than pow(y, x) and pairs are (2, 1), (2, 5) and (6, 1)
  - Input: X = {10, 19, 18}, Y = {11, 15, 9}; Output: 2, There are total 2 pairs where pow(x, y) is greater than pow(y, x) and pairs are (10, 11) and (10, 15)

Week #9

Objectives:

- To learn writing, executing and debugging programs related to Trees.

Outcomes:

- After completing this, the students would be able to develop, compile, debug, and fix errors of C programs based on Trees.

LAs:

- WAP Write the following menu driven program for the binary search tree

  ---
  Binary Search Tree Menu
  ---
  0. Quit
  1. Create
  2. In-Order Traversal
  3. Pre-Order Traversal
  4. Post-Order traversal
  5. Search
  6. Find Smallest Element
  7. Find Largest Element
  8. Deletion of Tree

  Enter your choice:

HAs:

- Extend the LA 1 by providing more options as follows:
  a) To count number of leaf nodes in the tree.
  b) To count number of non-leaf nodes in the tree.
  c) To find number of nodes in the tree.
  d) To find sum of all nodes of the tree.
  e) To print depth of the tree.
  f) To find nodes which are at maximum depth in the tree?
  g) To print all the elements of kth level in single line.
  h) To find the common ancestor and print the paths.
  i) To check whether a tree is a binary search tree or not.

Week #10

Objectives:

- To learn writing, executing and debugging programs related to graphs.

Outcomes:

- After completing this, the students would be able to develop, compile, debug, and fix errors of C programs based on graphs.

LAs:

- WAP to create an un-directed graph using adjacency matrix method.
- Modify the above program to include a menu driven program and add options for the depth-first traversal and breadth-first traversal.
- WAP to create a directed graph using adjacency list method.
- WAP to check whether an undirected graph is connected or not using DFS.
- WAP to check if an undirected graph is a tree or not using BFS.

HAs:

- Write a menu driven program to create an un-directed graph using Adjacency List Method and perform graph traversal operations.
- Write a menu driven program to create a directed graph using Adjacency List Method and perform graph traversal operations.
- WAP to check whether a directed graph is connected or not using DFS.
- WAP to check if a directed graph is a tree or not using DFS.