# Data and File Structure Laboratory Manual

Table of Contents

1. Array and String…….…………………………………………….…………………….. 6
2. Sorting and Searching.…………………………………………………………………...9
3. Stack….………………………………………………………………………………….10
4. Queue………………..…………………………………………………………………...12
5. Single Linked List…..……………………………………………………………………14
6. Double Linked List...…………………………………………………………………….16
7. Tree-1….……..…………………………………………………………………………..17
8. Tree-2.………………………………………………...………………………………….19
9. Graph……………………….…………………………………………………………….20
10. Hashing………………………………… ……………………………………………….21

# Rules:

1. The above mentioned 10 laboratories must be completed in a semester.
2. Each laboratory must be completed in two labs (One practice lab followed by an evaluation lab). In practice lab the lab assistants, faculty will assist the student whereas in evaluation lab no assistance from any of the staffs will be entertained.
3. Each laboratory (Excluding Laboratory-0) carries 100 marks.
   1. Attending practice lab……………….20 Marks

Continuous Evaluation

* 1. Record………………………………..20 Marks
  2. Evaluation Lab……………………….30 Marks
  3. Lab Test..……………………………..30 Marks

One-time Evaluation at the end of 10 laboratories

* + 1. Programming Ability Test……20 Marks
    2. Viva/Quiz Test………………...10 Marks

1. All practice lab and evaluation lab questions will be asked from the lab manual.
2. Absenteeism in any lab means corresponding marks will be deducted.
3. Discipline has the highest priority.

Laboratory-1

**Array and String**

1. Write a program that accepts all elements of an integer array and finds out sum and mean and standard deviation among the values.
2. Write a program that accepts all elements of a double array and finds its maximum and next maximum elements without using sorting.
3. Write a program that accepts a string and count the total no of capital letters, small letters and digits present in it.
4. Write a program to take two string inputs and find out whether they are same or different.
5. Write a program to check whether two strings are anagrams. The strings are said to be anagrams of each other if the letters from one sting can be rearranged to form the other string. From the above definition it is clear that two strings are anagrams if all characters in both strings occur same number of times.
6. Write a program to take a string and print it in lower case, uppercase, Sentence case and toggle case.
7. Write a program that concatenates two strings. The second string is concatenated after the first string and it becomes the resultant string.
8. Write a program to check whether a string is a palindrome or not without using extra space.
9. Write a program to accept an array of integers. Allow user to insert an new element as desired position or delete an element from any position. Note: Use user defined functions for insertion and deletion.
10. Write a C program to accept a matrix of order M x N and store its elements and interchange the main diagonal elements of the matrix with that of the secondary diagonal elements. Note: Allocate the 2-D array dynamically.
11. Write a C program to accept a matrix of order M x N and find the sum of each row and each column of a matrix.
12. Write a C program to accept a matric and determine whether it is a sparse matrix. A sparse matrix is matrix which has more zero elements than nonzero elements.
13. Write a program that generates all possible permutation with few characters.[if a,b,c are given then it would generate all 6 possible permutations]
14. Write a program to add two numbers where the maximum number of digits in each number is 500.
15. Write a program to multiply two numbers where the maximum number of digits in each number is 500.
16. Write a program to dynamically allocate an array and reverse the array using a user defined function reverseArray( ) without using any extra space. Note: Allocate the array within the user defined function, reverse it and return the array to the calling function.
17. Write a program to add two matrices using a user defined function addMatrix( ) which returns the dynamically allocated resultant matrix.
18. Write a function to take two metrics as argument (using pointers) and return the result of their multiplication as pointer to the resultant array.

Laboratory-2

**Sorting and Searching**

1. Write a program to create an array dynamically, accept its members and sort the array using following sorting algorithm. Also count the total number of swaps.
   1. Bubble sort
   2. Selection sort
   3. Insertion sort
   4. Quick sort
   5. Merge sort
2. Write a function search an element from the array using following searching techniques:
   1. Linear search
   2. Recursive linear search
   3. Binary search
   4. Recursive binary search
   5. Ternary search
3. A Fibonacci search searches for a key in a sorted array using divide and conquer rule but doesn’t use any division operation in practical. Implement Fibonacci search to search an element in a sorted array.
4. Write program to sort an array of double numbers based on their fractional part values.

**Sample Input**

**12.5 4.8 13.2 2.7 1.6**

**Sample Output**

**13.2 12.5 1.6 2.7 4.8**

1. Inversion Count for an array indicates – how far (or close) the array is from being sorted. If array is already sorted then inversion count is 0. If array is sorted in reverse order that inversion count is the maximum. Formally speaking, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j. Write a program to count inversions in an array.

**Sample Input**

**2 3 8 6 1**

**Sample Output**

**5**

**Explanation**

*Case 2:* The inversions are (2, 1), (3, 1), (8, 6), (8, 1), (6, 1)

Laboratory-3

**Stack**

1. Write a structure for an integer stack, implement **push, pop, and peek, IsEmpty** and **IsFull** function. Write a main function and call the functions based on user’s choice.

typedef struct stack

{

int top;

int data[max];

}Stack;

1. Implement a stack of characters and create mystack.h. Write a program to check whether an entered string is a palindrome or not. Include mystack.h for doing this.
2. Including mystack.h as a header file into your program, reverse a string.
3. Including mystack.h write a program to convert an infix expression into its equivalent postfix expression.
4. Including mystack.h write a program to evaluate a postfix expression.
5. Including mystack.h write a program to check whether the parenthesis of a C file is balanced or not.

Laboratory-4

**Queue**

1. Write a structure for an integer queue, implement **enqueue, dequeue, and traverse, IsEmpty** and **IsFull** function. Write a main function and call the functions based on user’s choice.

typedef struct queue

{

int front,rear;

int data[max];

}Queue;

1. Write a program to implement queue using two stacks. Include mystack.h and do the program.
2. Write a structure for an integer circular queue, implement **enqueue, dequeue, and traverse, IsEmpty** and **IsFull** function. Write a main function and call the functions based on user’s choice.

typedef struct circularQueue

{

int front,rear;

int data[max];

}Queue;

1. Defining structure of a circular queue (with a counter) , write functions for inserting, deleting and counting no of elements present in the queue. Write functions IsFull and IsEmpty also. Write main function to call them.
2. Write a program to implement a Deque or Double Ended Queue. Write functions for **insetFront(), insertRear(), deleteFront(), deleteRear()**.
3. Write a program to implement a priority queue where each element is associated with a priority value. Implement the functions insert(), getHighestPriority(), deleteHighestPriority().

Laboratory-5

**Single Linked List**

1. Create a singly linked list of integers, write functions to add elements at different places (beginning, end, at a specified position), delete a node from different positions (beginning, end, at a specified position) and traverse the linked list based on user’s choice.
2. Write a program to implement stack using linked list.
3. Write a program to implement Queue using linked list.
4. Write a function to reverse a single linked list without allocating any extra storage. Test the function by writing the function in the solution of question-1.
5. Write a function to find the mid-element of the linked list.
6. Implement insertion sort using single linked list.
7. Write a recursive and non-recursive functions to count the number of nodes in a linked list.
8. Write a function to delete the nth node from the end of a linked list.
9. Write a function to find sum of all nodes of a linked list created in question-1.
10. Write a function to pair-wise swap the nodes of a linked list.
11. Write a function to remove duplicate elements of a sorted linked list.
12. Write a function to merge two linked lists.
13. Write a function that counts the number of times a given int occurs in a Linked List.
14. Write a function to check whether the created linked list (in question-1) is palindrome or not.
15. Write a function that deletes each even numbered node from a null terminated singly list. Write function to print the list in a recursive manner.
16. Represent a polynomial of a single variable using a singly linked list. Write functions createPolynomial that stores one polynomial in a singly linked list. Write a function to add two such linked lists. Display the result in a flat file in an understandable manner.
17. Write a function to multiply two polynomials of a single variable using linked lists.
18. Write a function to check whether a linked list contain a cycle or not.

Laboratory-6

**Circular & Double Linked List**

1. Create a singly circular linked list of integers, write functions to add elements at different places (beginning, end, at a specified position), delete a node from different positions (beginning, end, at a specified position) and traverse the linked list based on user’s choice.
2. Create a doubly linked list of characters, write functions to add elements at different places (beginning, end, at a specified position), delete a node from different positions (beginning, end, at a specified position) and traverse the linked list in both directions based on user’s choice.
3. Create a doubly circular linked list of characters, write functions to add elements at different places (beginning, end, at a specified position), delete a node from different positions (beginning, end, at a specified position) and traverse the linked list based on user’s choice.
4. Write a function for doubly linked list to check whether the contents are palindrome or not.
5. Given a doubly linked list, rotate the linked list counter-clockwise by k nodes.

# An ordinary Doubly Linked List requires space for two address fields to store the addresses of previous and next nodes. A memory efficient version of Doubly Linked List can be created using only one space for address field with every node. This memory efficient Doubly Linked List is called XOR Linked List or Memory Efficient as the list uses bitwise XOR operation to save space for one address. In the XOR linked list, instead of storing actual memory addresses, every node stores the XOR of addresses of previous and next nodes. Write a program to implement XOR Linked List.

Laboratory-7

**TREE-I**

1. Declare a binary search tree (BST) where information at each node would be a single integer. Write recursive and non recursive (use mystack.h) functions for
   1. Inserting a key
   2. Deleting a key from the tree.
   3. Searching an element
   4. Inorder , Preorder and Postorder traversal
   5. Finding height of the tree
   6. Count number of nodes
   7. Display leaf nodes
   8. Count number of non-leaf nodes
   9. Find sum of all nodes
   10. Find minimum value of the tree
   11. Find maximum value of the tree
2. Write a program to construct a BST from pre-order traversal.
3. Write a program to find a pair of nodes with a given sum in a BST.
4. Write a program to print the common nodes in two BSTs.
5. Write a program to check whether a given binary tree is a binary search tree or not.
6. Write a program to check if each internal node of a BST has exactly one child.
7. Write a program to check if there is a triplet in a Balanced BST that adds to zero.
8. Write a program to construct a tree from its pre-order and in-order traversal.
9. Write a program to construct a tree from its post-order and in-order traversal.

Laboratory-8

**TREE-II**

1. Declare an AVL Tree where information at each node would be a single integer. Write recursive functions for
   1. Inserting a key
   2. Deleting a key from the tree.
   3. Searching an element
   4. Inorder , Preorder and Postorder traversal
   5. Count number of nodes
   6. Display non-leaf nodes
   7. Count number of leaf nodes
   8. Find minimum value of the tree
   9. Find maximum value of the tree
2. Write a program to implement single threaded binary tree and perform the following functions.
   1. Inserting a key
   2. Deletion of a key
   3. In-order traversal using the thread
   4. Maximum depth of the tree
3. Write a program to check whether the given binary tree is heap or not.
4. Write a program to implement ternary search tree.
5. Write a function to check whether all the leaves of a AVL-tree are at same level or not.
6. Write a function to print all nodes at distance k from the leaf in an AVL tree.

Laboratory-9

**Graph**

# Write a program for Breadth First Traversal of a graph.

# Write a program for Depth First Traversal of a graph.

1. Write a program to check whether a graph contains a cycle or not.
2. A directed graph is called strongly connected if there is a path from each vertex in the graph to every other vertex. Write a program to check whether a graph is Strongly Connected or not.
3. Write a program to check whether there is a path between two vertices of graph.
4. Given a directed graph. Write a program to find the transitive closure of the graph.
5. Given a directed graph. Write a program to find shortest path among all the nodes of a graph using Floyd Warshall Algorithm.
6. Given an undirected, connected and weighted graph, find **M**inimum **S**panning **T**ree (MST) of the graph using Kruskal’s Algorithm.
7. Given an undirected, connected and weighted graph, find **M**inimum **S**panning **T**ree (MST) of the graph using Prim’s Algorithm.

Laboratory-10

**Hashing**