VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

DATA STRUCTURES

Submitted by

SAMRITH SANJOO.S (1BM21CS185)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Oct 2022-Feb 2023

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**DATA STRUCTURES**" carried out by **SAMRITH SANJOO.S (1BM21CS185),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022-23. The Lab report has been approved as it satisfies the academic requirements in respect of Data structures Lab **- (22CS3PCDST) work** prescribed for the said degree.

Dr. Selva Kumar S
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

Index Sheet

Sl. No.	Experiment Title	Page No.
1	Write a program to simulate the working of stack using an array with the following: a) Push b) Pop c) Display The program should print appropriate messages for stack overflow, stack underflow.	1-3
2	WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide).	4-6
3	WAP to simulate the working of a queue of integers using an array. Provide the following operations: a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions	7-10
4	WAP to simulate the working of a circular queue of integers using an array. Provide the following operations: a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions	11-14

5	Part 1: WAP to Implement Singly Linked List with following operations: a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list.	15-23
6	Part 2: WAP to Implement Singly Linked List with following operations a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.	15-23
7	WAP Implement Single Link List with following operations: a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists	24-31
8	Write a program to implement Stacks and Queues using a linked list.	32-37
9	WAP Implement doubly link list with primitive operations: a) Create a doubly linked list. b) Insert a new node to the left of the node. c) Delete the node based on a specific value d) Display the contents of the list	38-43
10	Write a program a) To construct a binary Search tree. b) To traverse the tree using all the methods i.e., in-order, preorder and postorder c) To display the elements in the tree.	44-46

Course Outcome

CO1	Apply the concept of linear and nonlinear data structures.
CO2	Analyze data structure operations for a given problem.
CO3	Design and develop solutions using the operations of linear and nonlinear data structure for a given specification.
CO4	Conduct practical experiments for demonstrating the operations of different data structures.

LAB PROGRAM 1: Write a program to simulate the working of stack using an array with the following:

a) Push

b) Pop

c) Display

The program should print appropriate messages for stack overflow, stack underflow.

Program :-

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 100

void push(int stack[], int *top, int *ptr)
{
    if (*top == SIZE)
    {
        printf("Stack Overflow!");
    }
    else
    {
        stack[++(*top)] = *ptr;
    }
}

int pop(int *top, int stack[])
{
    int del_item;
    if (*top == -1)
    {
        printf("\nStack Underflow!");
    }
    else
    {
        del_item = stack[*top];
        (*top)--;
        return del_item;
    }
}
```

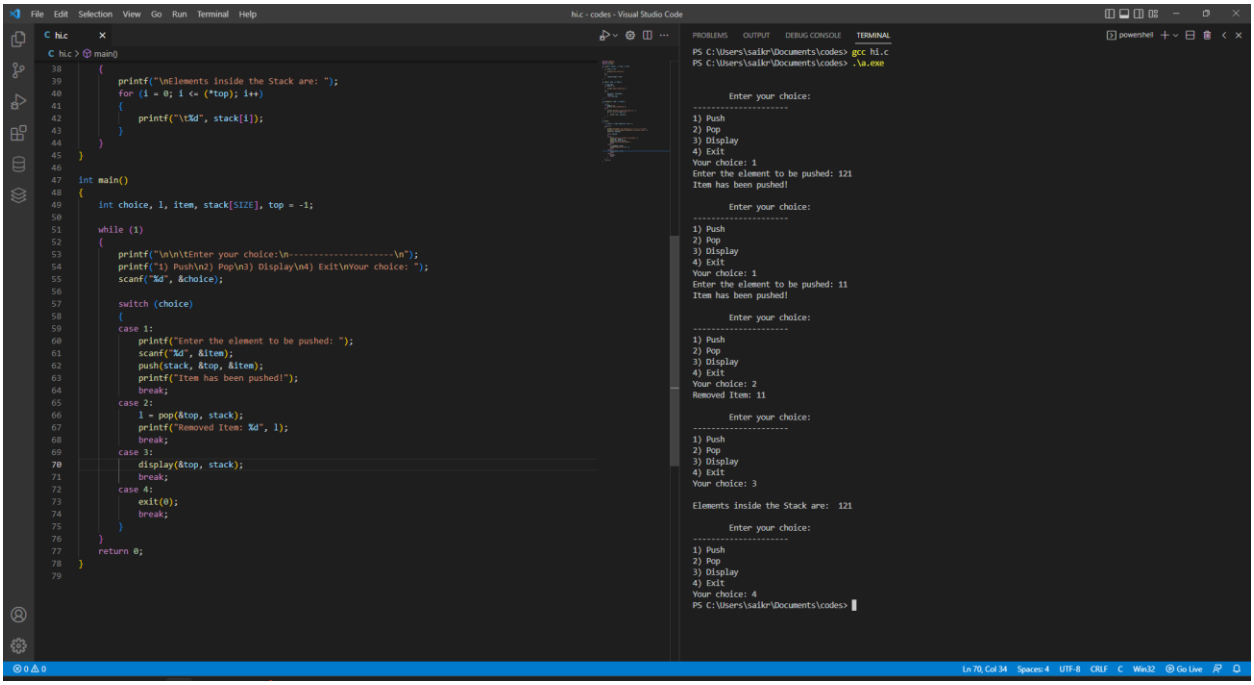
```c
void display(int *top, int stack[])
{
    int i;
    if (*top == -1)
        printf("\nStack Underflow!");
    else
    {
        printf("\nElements inside the Stack are: ");
        for (i = 0; i <= (*top); i++)
        {
            printf("\t%d", stack[i]);
        }
    }
}

int main()
{
    int choice, l, item, stack[SIZE], top = -1;

    while (1)
    {
        printf("\n\n\tEnter your choice:\n---------------------\n");
        printf("1) Push\n2) Pop\n3) Display\n4) Exit\nYour choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("Enter the element to be pushed: ");
            scanf("%d", &item);
            push(stack, &top, &item);
            printf("Item has been pushed!");
            break;
        case 2:
            l = pop(&top, stack);
            printf("Removed Item: %d", l);
            break;
        case 3:
            display(&top, stack);
            break;
        case 4:
            exit(0);
            break;
        }
    }
    return 0;
}
```

Output :-



Lab Program 2: WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide).

Program :-

```
#include <stdio.h>
#include <string.h>

int top = -1;
char s[20];
char infix[20];
char postfix[20];

int sp(char item)
{
    switch (item)
    {
    case '+':
    case '-':
        return 2;
    case '*':
    case '/':
        return 4;
    case '^':
    case '$':
        return 5;
    case '(':
        return 0;
    case '#':
        return -1;
    default:
        return 8;
    }
}

int ip(char item)
{
    switch (item)
    {
    case '+':
    case '-':
        return 1;
    case '*':
    case '/':
        return 3;
    case '^':
    case '$':
        return 6;
    case '(':
        return 9;
```

```
    case ')':
        return 0;
    default:
        return 7;
    }
}

void push(char item)
{
    s[++top] = item;
}

char pop()
{
    return s[top--];
}

void inf_to_post()
{
    int i, j = 0;
    char symbol;
    push('#');
    for (i = 0; i < strlen(infix); i++)
    {
        symbol = infix[i];

        while (sp(s[top]) > ip(symbol))
        {
            postfix[j] = pop();
            j++;
        }

        if (sp(s[top]) < ip(symbol))
        {
            push(symbol);
        }

        if (sp(s[top]) == ip(symbol))
        {
            pop();
        }
    }

    while (s[top] != '#')
    {
        postfix[j] = pop();
        j++;
    }
    postfix[j] = '\0';
```

```c
}

int main()
{
    printf("Enter a valid infix expression: ");
    scanf("%s", infix);
    inf_to_post();
    printf("The corresponding postfix expression is: %s", postfix);
    return 0;
}
```

Output :-

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
PS C:\Users\saikr\Documents\codes> gcc hi.c
PS C:\Users\saikr\Documents\codes> .\a.exe
Enter a valid infix expression: (a+b)/(b+c)+e+c
The corresponding postfix expression is: ab+bc+/e+c+
PS C:\Users\saikr\Documents\codes> gcc hi.c
PS C:\Users\saikr\Documents\codes> .\a.exe
Enter a valid infix expression: a+b-c+(c*d)/d
The corresponding postfix expression is: ab+c-cd*d/+
PS C:\Users\saikr\Documents\codes>
```

Lab Program 3: WAP to simulate the working of a queue of integers using an array. Provide the following operations:

a) Insert

b) Delete

c) Display

The program should print appropriate messages for queue empty and queue overflow conditions

---

Program :-

```c
#include <stdio.h>
```

```c
#include <stdlib.h>
#define size 5

void insert(int *q, int *r, int *item)
{
    if ((*r) == (size - 1))
    {
        printf("Queue Overflow!\n");
    }
    else
    {
        (*r)++;
        q[(*r)] = (*item);
    }
}

void delete_front(int *q, int *r, int *f)
{
    if ((*r) < (*f))
    {
        printf("Queue Underflow!\n");
    }
    else
    {
        printf("The element deleted is : %d", q[(*f)++]);
    }
}

void display(int *q, int *r, int *f)
{
    if ((*r) < (*f))
    {
        printf("The Queue is empty\n");
    }
    else
    {
        printf("The elements in the queue are : ");
        int i;
        for (i = (*f); i <= (*r); i++)
        {
            printf("%d\t", q[i]);
        }
    }
}

int main()
{
    int q[size], item, i, r = -1, f = 0, choice;
    while (1)
```

```
    {
        printf("\n\nMain Menu\n----------\n1) Insert\n2) Delete\n3) Display\n----
------\nEnter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter the element to be inserted : ");
            scanf("%d", &item);
            insert(q, &r, &item);
            break;
        case 2:
            delete_front(q, &r, &f);
            break;
        case 3:
            display(q, &r, &f);
            break;
        default:
            exit(0);
        }
    }
    return 0;
}
```

Output :-

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
PS C:\Users\saikr\Documents\codes> gcc hi.c
PS C:\Users\saikr\Documents\codes> .\a.exe


Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 3
The Queue is empty


Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 2
Queue Underflow!


Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 1
Enter the element to be inserted : 11


Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 1
Enter the element to be inserted : 121


Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 2
The element deleted is : 11
```

```
Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 3
The elements in the queue are : 121

Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 1
Enter the element to be inserted : 11


Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 1
Enter the element to be inserted :
```

Lab Program 4: WAP to simulate the working of a circular queue of integers using an array. Provide the

following operations.

a) Insert

b) Delete

c) Display

The program should print appropriate messages for queue empty and queue overflow conditions

---

Program :-

```
#include <stdio.h>
#include <stdlib.h>
#define size 5

void insert(int *q, int *r, int *item, int *count)
{
    if ((*count) == size)
    {
        printf("Queue Overflow!\n");
    }
    else
    {
        (*r) = ((*r) + 1) % size;
        q[(*r)] = (*item);
        (*count)++;
    }
}

void delete_front(int *q, int *r, int *f, int *count)
{
    if ((*count) == 0)
    {
        printf("Queue Underflow!\n");
    }
    else
    {
        printf("The element deleted is : %d", q[*f]);
        (*f) = ((*f) + 1) % size;
        (*count)--;
```

```c
    }
}

void display(int *q, int *r, int *f, int *count)
{
    if ((*count) == 0)
    {
        printf("The Queue is empty\n");
    }
    else
    {
        printf("\nThe elements in the queue are : ");
        int i, front;
        front = (*f);
        for (i = 0; i < (*count); i++)
        {
            printf("%d\t", q[front]);
            front = (front + 1) % size;
        }
    }
}

int main()
{
    int q[size], item, i, r = -1, f = 0, choice, count = 0;
    while (1)
    {
        printf("\n\nMain Menu\n----------\n1) Insert\n2) Delete\n3) Display\n----
------\nEnter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter the element to be inserted : ");
            scanf("%d", &item);
            insert(q, &r, &item, &count);
            break;
        case 2:
            delete_front(q, &r, &f, &count);
            break;
        case 3:
            display(q, &r, &f, &count);
            break;
        default:
            exit(0);
        }
    }
    return 0;
}
```

Output :-

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Users\saikr\Documents\codes> gcc hi.c
PS C:\Users\saikr\Documents\codes> .\a.exe


Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 1
Enter the element to be inserted : 121


Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 1
Enter the element to be inserted : 122


Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 1
Enter the element to be inserted : 223


Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 1
Enter the element to be inserted : 34


Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 1
Enter the element to be inserted : 324
```

```
Enter your choice : 1
Enter the element to be inserted : 1123
Queue Overflow!


Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 2
The element deleted is : 121

Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 2
The element deleted is : 122

Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 1
Enter the element to be inserted : 1131


Main Menu
----------
1) Insert
2) Delete
3) Display
----------
Enter your choice : 3

The elements in the queue are : 223     34      324     1131
```

Lab Program 5, 6: WAP to Implement Singly Linked List with following operations:

a) Create a linked list.

b) Insertion of a node at first position, at any position and at end of list.

c) Deletion of first element, specified element and last element in the list.

d) Display the contents of the linked list.

Program :-

```
#include <stdio.h>
#include <stdlib.h>

struct NODE
{
    int value;
    struct NODE *next;
};
typedef struct NODE *node;

node insert_at_beginning(int item, node first)
{
    node temp = (node)malloc(sizeof(struct NODE));
    if (temp == NULL)
    {
        printf("\nMemory not allocated!");
    }
    (temp->value) = item;
    (temp->next) = NULL;
    if (first == NULL)
    {
        return temp;
    }
    else
    {
        temp->next = first;
        first = temp;
        return first;
    }
}

node insert_at_end(int item, node first)
{
```

```
    node temp = (node)malloc(sizeof(struct NODE));
    if (temp == NULL)
    {
        printf("\nMemory not allocated!");
    }
    (temp->value) = item;
    (temp->next) = NULL;
    if ((first->next) == NULL)
    {
        (first->next) = temp;
        return first;
    }
    else
    {
        node last = first;
        while ((last->next) != NULL)
        {
            last = (last->next);
        }
        (last->next) = temp;
        return first;
    }
}

node insert_at_any_position(int item, int position, node first)
{
    node new, curr, prev;
    new = malloc(sizeof(struct NODE));
    int i = 1;
    (new->value) = item;
    (new->next) = NULL;
    if (first == NULL && position == 1)
    {
        return new;
    }
    else
    {
        prev = NULL;
        curr = first;
        while ((i != position) && (curr != NULL))
        {
            prev = curr;
            curr = (curr->next);
            i++;
        }
        if (i == position)
        {
            prev->next = new;
            new->next = curr;
```

```
            return first;
        }
        else if (curr == NULL)
        {
            printf("\nPosition not found!");
            return first;
        }
        else if (first != NULL && position == 1)
        {
            return insert_at_beginning(item, first);
        }
    }
}

node delete_at_the_beginning(node first)
{
    if (first == NULL)
    {
        printf("\nCannot delete, the Linked List is empty");
        return NULL;
    }
    else
    {
        node temp;
        temp = first;
        first = (first->next);
        free(temp);
        return first;
    }
}

node delete_at_the_end(node first)
{
    if (first == NULL)
    {
        printf("\nCannot delete, the Linked List is empty");
        return NULL;
    }
    else
    {
        node prev, curr;
        prev = NULL;
        curr = first;
        while ((curr->next) != NULL)
        {
            prev = curr;
            curr = (curr->next);
        }
        (prev->next) = NULL;
```

```
        free(curr);
        return first;
    }
}

node delete_at_any_position(int pos, node first)
{
    if (first == NULL)
    {
        printf("The linked list is empty!");
        return NULL;
    }
    else if (first->next == NULL)
    {
        if (pos == 1)
            return NULL;
        else
        {
            printf("Position not found!");
            return NULL;
        }
    }
    else
    {
        int count = 0;
        node prev = NULL, curr = first;
        while ((count != pos) && (curr != NULL))
        {
            prev = curr;
            curr = (curr->next);
            count++;
        }
        if (count == pos)
        {
            prev->next = curr->next;
            free(curr);
            return first;
        }
        else
        {
            printf("\nPosition not found!");
            return first;
        }
    }
}

void display(node first)
{
    node temp;
```

```
    temp = first;
    if (temp == NULL)
    {
        printf("\nThe Linked list is empty!");
    }
    else
    {
        printf("The elements in the node are : ");
        while (temp != NULL)
        {
            printf("%d\t", (temp->value));
            temp = (temp->next);
        }
    }
}

int main()
{
    int choice, pos, item, x;
    node first = NULL;
    while (1)
    {
        printf("\n\nMenu\n----------------------------------------\n1) Insert at
beginning\n2) Insert at end\n3) Insert at any position\n4) Delete at
beginning\n5) Delete at end\n6) Delete at any position\n7) Display\n-------------
---------------------------\nEnter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter the element to be inserted : ");
            scanf("%d", &x);
            first = insert_at_beginning(x, first);
            break;
        case 2:
            printf("Enter the element to be inserted : ");
            scanf("%d", &x);
            first = insert_at_end(x, first);
            break;
        case 3:
            printf("Enter the position : ");
            scanf("%d", &pos);
            printf("Enter the element to be inserted : ");
            scanf("%d", &x);
            first = insert_at_any_position(x, pos, first);
            break;
        case 4:
            first = delete_at_the_beginning(first);
            break;
```

```
        case 5:
            first = delete_at_the_end(first);
            break;
        case 6:
            printf("Enter the position where the element is to be deleted : ");
            scanf("%d", &pos);
            first = delete_at_any_position(pos, first);
            break;
        case 7:
            display(first);
            break;
        default:
            exit(0);
            break;
        }
    }
    return 0;
}
```

Output :-

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
PS C:\Users\saikr\Documents\codes> gcc hi.c
PS C:\Users\saikr\Documents\codes> .\a.exe


Menu
----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Delete at beginning
5) Delete at end
6) Delete at any position
7) Display
----------------------------------------
Enter your choice : 1
Enter the element to be inserted : 11


Menu
----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Delete at beginning
5) Delete at end
6) Delete at any position
7) Display
----------------------------------------
Enter your choice : 2
Enter the element to be inserted : 22


Menu
----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Delete at beginning
5) Delete at end
6) Delete at any position
7) Display
----------------------------------------
Enter your choice : 3
Enter the position : 2
Enter the element to be inserted : 33
```

```
Menu
----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Delete at beginning
5) Delete at end
6) Delete at any position
7) Display
----------------------------------------
Enter your choice : 7
The elements in the node are : 11       33      22

Menu
----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Delete at beginning
5) Delete at end
6) Delete at any position
7) Display
----------------------------------------
Enter your choice : 4


Menu
----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Delete at beginning
5) Delete at end
6) Delete at any position
7) Display
----------------------------------------
Enter your choice : 7
The elements in the node are : 33       22

Menu
----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Delete at beginning
5) Delete at end
6) Delete at any position
7) Display
----------------------------------------
Enter your choice : 5
```

```
Menu
-----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Delete at beginning
5) Delete at end
6) Delete at any position
7) Display
-----------------------------------------
Enter your choice : 7
The elements in the node are : 33

Menu
-----------------------------------------
1) Insert at beginning
2) Insert at end
3) Insert at any position
4) Delete at beginning
5) Delete at end
6) Delete at any position
7) Display
-----------------------------------------
Enter your choice : 6
Enter the position where the element is to be deleted : 44
Position not found!
```

Lab Program 7: WAP Implement Single Link List with following operations:

a) Sort the linked list.

b) Reverse the linked list.

c) Concatenation of two linked lists

Program :-

```
#include <stdio.h>
#include <stdlib.h>

struct NODE
{
    int value;
    struct NODE *next;
};
typedef struct NODE *node;

node insert_at_beginning(int item, node first)
{
    node temp = (node)malloc(sizeof(struct NODE));
    if (temp == NULL)
    {
        printf("\nMemory not allocated!");
    }
    (temp->value) = item;
    (temp->next) = NULL;
    if (first == NULL)
    {
        return temp;
    }
    else
    {
        temp->next = first;
        first = temp;
        return first;
    }
}

node delete_at_the_beginning(node first)
{
    if (first == NULL)
    {
        printf("Cannot delete, the Linked List is empty");
```

```
        return NULL;
    }
    else
    {
        node temp;
        temp = first;
        first = (first->next);
        free(temp);
        return first;
    }
}

node sort(node first)
{
    int temp;
    node curr = first;
    if (first == NULL)
    {
        printf("Linked list is empty!");
        return NULL;
    }
    else
    {
        while (curr->next != NULL)
        {
            node check = curr->next;
            while (check != NULL)
            {
                if (curr->value > check->value)
                {
                    temp = curr->value;
                    curr->value = check->value;
                    check->value = temp;
                }
                check = check->next;
            }
            curr = curr->next;
        }
        return first;
    }
}

node concatenate(node f1, node f2)
{
    if (f1 == NULL && f2 == NULL)
    {
        printf("The linked lists are empty!");
        return NULL;
    }
```

```
    else if (f1 != NULL && f2 == NULL)
        return f1;
    else if (f1 == NULL && f2 != NULL)
        return f2;
    else
    {
        node last = f1;
        while (last->next != NULL)
        {
            last = last->next;
        }
        last->next = f2;
        return f1;
    }
}

node reverse(node first)
{
    if (first == NULL)
    {
        printf("The linked lists are empty!");
        return NULL;
    }
    else
    {
        node rev = NULL;
        while (first != NULL)
        {
            node Next = first->next;
            first->next = rev;
            rev = first;
            first = Next;
        }
        return rev;
    }
}

void display(node first)
{
    node temp;
    temp = first;
    if (temp == NULL)
    {
        printf("The Linked list is empty!");
    }
    else
    {
        printf("The elements in the node are :  ");
        while (temp != NULL)
```

```
        {
            printf("%d  ", (temp->value));
            temp = (temp->next);
        }
    }
}

int main()
{
    int choice, n, i, val, x;
    node first = NULL, f1 = NULL, f2 = NULL;
    while (1)
    {
        printf("\n\nEnter the operations to be performed :\n1) Push\n2) Pop\n3)
Sort\n4) Concatenate\n5) Reverse\n6) Display\nEnter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter the element to be inserted : ");
            scanf("%d", &x);
            first = insert_at_beginning(x, first);
            break;
        case 2:
            first = delete_at_the_beginning(first);
            break;
        case 3:
            first = sort(first);
            break;
        case 4:

            printf("Enter the number of fields for linked list 1 : ");
            scanf("%d", &n);
            printf("Enter %d entries : ", n);
            for (i = 0; i < n; i++)
            {
                scanf("%d", &val);
                f1 = insert_at_beginning(val, f1);
            }
            printf("Enter the number of fields for linked list 2 : ");
            scanf("%d", &n);
            printf("Enter %d entries : ", n);
            for (i = 0; i < n; i++)
            {
                scanf("%d", &val);
                f2 = insert_at_beginning(val, f2);
            }
            printf("The concatenated linked list is : ");
            f1 = concatenate(f1, f2);
```

```
            display(f1);
            break;
        case 5:
            first = reverse(first);
            break;
        case 6:
            display(first);
            break;
        default:
            exit(0);
        }
    }
    return 0;
}
```

Output :-

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
PS C:\Users\saikr\Documents\codes> gcc hi.c
PS C:\Users\saikr\Documents\codes> .\a.exe


Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 1
Enter the element to be inserted : 121


Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 1
Enter the element to be inserted : 11


Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 1
Enter the element to be inserted : 22


Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 1
Enter the element to be inserted : 45
```

```
Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 6
The elements in the node are :  45  22  11  121

Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 5


Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 6
The elements in the node are :  121  11  22  45
```

```
Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 6
The elements in the node are :  121  11  22  45

Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 3


Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 6
The elements in the node are :  11  22  45  121

Enter the operations to be performed :
1) Push
2) Pop
3) Sort
4) Concatenate
5) Reverse
6) Display
Enter your choice : 4
Enter the number of fields for linked list 1 : 3
Enter 3 entries : 1 2 3
Enter the number of fields for linked list 2 : 5
Enter 5 entries : 2 33 4 55 6 7
The concatenated linked list is : The elements in the node are :  3  2  1  6  55  4  33  2
```

Lab Program 8: Write a program to implement Stacks and Queues using a linked list.

Program :-

```
#include <stdio.h>
#include <stdlib.h>

struct NODE
{
    int value;
    struct NODE *next;
};
typedef struct NODE *node;

node insert_at_beginning(int item, node first)
{
    node temp = (node)malloc(sizeof(struct NODE));
    if (temp == NULL)
    {
        printf("\nMemory not allocated!");
    }
    (temp->value) = item;
    (temp->next) = NULL;
    if (first == NULL)
    {
        return temp;
    }
    else
    {
        temp->next = first;
        first = temp;
        return first;
    }
}

node delete_at_the_beginning(node first)
{
    if (first == NULL)
    {
        printf("\nCannot delete, the Linked List is empty");
        return NULL;
    }
    else
    {
```

```c
        node temp;
        temp = first;
        first = (first->next);
        free(temp);
        return first;
    }
}

node delete_at_the_end(node first)
{
    if (first == NULL)
    {
        printf("\nCannot delete, the Linked List is empty");
        return NULL;
    }
    else
    {
        node prev, curr;
        prev = NULL;
        curr = first;
        while ((curr->next) != NULL)
        {
            prev = curr;
            curr = (curr->next);
        }
        (prev->next) = NULL;
        free(curr);
        return first;
    }
}

void display(node first)
{
    node temp;
    temp = first;
    if (temp == NULL)
    {
        printf("\nThe Linked list is empty!");
    }
    else
    {
        printf("The elements in the node are : ");
        while (temp != NULL)
        {
            printf("%d\t", (temp->value));
            temp = (temp->next);
        }
    }
}
```

```c
int main()
{
    int choice, pos, item, x;
    node first = NULL;
    printf("Enter the data structure you would want to create :\n1) Stack\n2)
Queue\nYour choice : ");
    scanf("%d", &choice);
    if (choice == 1)
    {
        while (1)
        {
            printf("\n\nEnter the operations to be performed :\n1) Push\n2)
Pop\n3) Display\nEnter your choice : ");
            scanf("%d", &choice);
            switch (choice)
            {
            case 1:
                printf("Enter the element to be inserted : ");
                scanf("%d", &x);
                first = insert_at_beginning(x, first);
                break;
            case 2:
                first = delete_at_the_beginning(first);
                break;
            case 3:
                display(first);
                break;
            default:
                exit(0);
            }
        }
    }
    else if (choice == 2)
    {
        while (1)
        {
            printf("\n\nEnter the operations to be performed :\n1) Enqueue\n2)
Dequeue\n3) Display\nEnter your choice : ");
            scanf("%d", &choice);
            switch (choice)
            {
            case 1:
                printf("Enter the element to be inserted : ");
                scanf("%d", &x);
                first = insert_at_beginning(x, first);
                break;
            case 2:
                first = delete_at_the_end(first);
```

```
                break;
            case 3:
                display(first);
                break;
            default:
                exit(0);
            }
        }
    }
    else
    {
        printf("Enter a valid choice!");
    }
    return 0;
}
```

Output :-

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
PS C:\Users\saikr\Documents\codes> gcc hi.c
PS C:\Users\saikr\Documents\codes> .\a.exe
Enter the data structure you would want to create :
1) Stack
2) Queue
Your choice : 1


Enter the operations to be performed :
1) Push
2) Pop
3) Display
Enter your choice : 1
Enter the element to be inserted : 112


Enter the operations to be performed :
1) Push
2) Pop
3) Display
Enter your choice : 1
Enter the element to be inserted : 1321


Enter the operations to be performed :
1) Push
2) Pop
3) Display
Enter your choice : 3
The elements in the node are : 1321     112

Enter the operations to be performed :
1) Push
2) Pop
3) Display
Enter your choice : 2


Enter the operations to be performed :
1) Push
2) Pop
3) Display
Enter your choice : 3
The elements in the node are : 112
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
PS C:\Users\saikr\Documents\codes> gcc hi.c
PS C:\Users\saikr\Documents\codes> .\a.exe
Enter the data structure you would want to create :
1) Stack
2) Queue
Your choice : 2


Enter the operations to be performed :
1) Enqueue
2) Dequeue
3) Display
Enter your choice : 1
Enter the element to be inserted : 121


Enter the operations to be performed :
1) Enqueue
2) Dequeue
3) Display
Enter your choice : 1
Enter the element to be inserted : 234


Enter the operations to be performed :
1) Enqueue
2) Dequeue
3) Display
Enter your choice : 2


Enter the operations to be performed :
1) Enqueue
2) Dequeue
3) Display
Enter your choice : 3
The elements in the node are : 234
```

Lab Program 9: WAP Implement doubly link list with primitive operations:

a) Create a doubly linked list.

b) Insert a new node to the left of the node.

c) Delete the node based on a specific value

d) Display the contents of the list

Program :-

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int value;
    struct node *next;
    struct node *prev;
};

typedef struct node *NODE;

NODE getnode()
{
    NODE temp;
    temp = (NODE)malloc(sizeof(struct node));

    if (temp == NULL)
    {
        printf("Memory could not be allocated.");
    }

    return (temp);
}

NODE insert_left(NODE first, int pos, int item)
{
    int count = 1;
    NODE New, curr;
    New = getnode();
    New->value = item;
    New->next = NULL;
    New->prev = NULL;
```

```c
    if (first == NULL)
    {
        return New;
    }

    if (pos == 1)
    {
        New->next = first;
        first->prev = New;
        first = New;
        return first;
    }

    curr = first;
    while (count != pos && curr->next != NULL)
    {
        curr = curr->next;
        count++;
    }

    if (count == pos)
    {
        New->next = curr;
        New->prev = curr->prev;
        (curr->prev)->next = New;
        curr->prev = New;
        printf("Item has been inserted");
        return first;
    }

    printf("Position couldnt be found");
    return first;
}

NODE del_spec(NODE first, int key)
{
    NODE temp, curr;

    if (first == NULL)
    {
        printf("Nothing to delete :(");
        return NULL;
    }

    if (key == first->value)
    {
        first = first->next;
        return first;
    }
```

```c
    curr = first;

    while (curr->next != NULL && curr->value != key)
    {
        curr = curr->next;
    }

    if (curr->next == NULL && curr->value == key)
    {
        (curr->prev)->next = curr->next;
        printf("Deleted Value:%d", curr->value);
        free(curr);
        return first;
    }

    if (curr->value == key)
    {
        (curr->prev)->next = curr->next;
        (curr->next)->prev = curr->prev;

        printf("Deleted Value:%d", curr->value);
        free(curr);
        return first;
    }

    printf("Couldn't find Value :(");
    return first;
}

void display(NODE first)
{
    NODE temp;
    temp = first;

    if (first == NULL)
    {
        printf("Whoops List is empty!");
    }

    while (temp != NULL)
    {
        printf(" %d", temp->value);
        temp = temp->next;
    }
}

int main()
{
```

```
    int choice, item, x;
    NODE first = NULL;
    while (1)
    {
        printf("\n\nMenu\n----------------------------------------\n1) Insert at
pos\n2) Del specific value\n3) Display\n4) Exit\n--------------------------------
---------\nEnter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter the element to be inserted : ");
            scanf("%d", &item);
            printf("Enter the position to be inserted : ");
            scanf("%d", &x);
            first = insert_left(first, x, item);
            break;
        case 2:
            printf("Enter the element to be deleted : ");
            scanf("%d", &item);
            first = del_spec(first, item);
            break;
        case 3:
            display(first);
            break;
        case 4:
            printf("Exiting...");
            exit(0);
            break;
        default:
            printf("Please enter correct choice :(");
            break;
        }
    }
    return 0;
}
```

Output :-

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
PS C:\Users\saikr\Documents\codes> gcc hi.c
PS C:\Users\saikr\Documents\codes> .\a.exe


Menu
----------------------------------------
1) Insert at pos
2) Del specific value
3) Display
4) Exit
----------------------------------------
Enter your choice : 1
Enter the element to be inserted : 121
Enter the position to be inserted : 1


Menu
----------------------------------------
1) Insert at pos
2) Del specific value
3) Display
4) Exit
----------------------------------------
Enter your choice : 1
Enter the element to be inserted : 34
Enter the position to be inserted : 1


Menu
----------------------------------------
1) Insert at pos
2) Del specific value
3) Display
4) Exit
----------------------------------------
Enter your choice : 1
Enter the element to be inserted : 234
Enter the position to be inserted : 1


Menu
----------------------------------------
1) Insert at pos
2) Del specific value
3) Display
4) Exit
----------------------------------------
Enter your choice : 3
 234 34 121
```

```
Menu
----------------------------------------
1) Insert at pos
2) Del specific value
3) Display
4) Exit
----------------------------------------
Enter your choice : 2
Enter the element to be deleted : 121
Deleted Value:121

Menu
----------------------------------------
1) Insert at pos
2) Del specific value
3) Display
4) Exit
----------------------------------------
Enter your choice : 3
 234 34
```

Lab Program 10: Write a program

a) To construct a binary Search tree.

b) To traverse the tree using all the methods i.e., in-order, preorder and postorder

c) To display the elements in the tree.

Program :-

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int data;
    struct node *left;
    struct node *right;
};

struct node *insert(struct node *node, int data)
{
    if (node == NULL)
    {
        struct node *temp = (struct node *)malloc(sizeof(struct node));
        temp->data = data;
        temp->left = temp->right = NULL;
        return temp;
    }

    if (data < node->data)
        node->left = insert(node->left, data);
    else if (data > node->data)
        node->right = insert(node->right, data);

    return node;
}

void inorder(struct node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d ", root->data);
```

```c
        inorder(root->right);
    }
}

void preorder(struct node *root)
{
    if (root != NULL)
    {
        printf("%d ", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

void postorder(struct node *root)
{
    if (root != NULL)
    {
        postorder(root->left);
        postorder(root->right);
        printf("%d ", root->data);
    }
}

int main()
{
    struct node *root = NULL;
    int n, i, element;

    printf("Enter the number of elements to be inserted: ");
    scanf("%d", &n);
    printf("Enter %d elements: ", n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &element);
        root = insert(root, element);
    }

    printf("In-order traversal: ");
    inorder(root);
    printf("\nPre-order traversal: ");
    preorder(root);
    printf("\nPost-order traversal: ");
    postorder(root);

    return 0;
}
```

Output :-

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
PS C:\Users\saikr\Documents\codes> gcc hi.c
PS C:\Users\saikr\Documents\codes> .\a.exe
Enter the number of elements to be inserted: 5
Enter 5 elements: 23 33 3 1 78
In-order traversal: 1 3 23 33 78
Pre-order traversal: 23 3 1 33 78
Post-order traversal: 1 3 78 33 23
PS C:\Users\saikr\Documents\codes> .\a.exe
Enter the number of elements to be inserted: 7
Enter 7 elements: 1 99 3 55 383 223 412
In-order traversal: 1 3 55 99 223 383 412
Pre-order traversal: 1 99 3 55 383 223 412
Post-order traversal: 55 3 223 412 383 99 1
PS C:\Users\saikr\Documents\codes>
```