

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



## DATA STRUCTURE LAB RECORD

*Submitted by*

**S SHREE LAKSHMI (1BM19CS136)**

*Under the Guidance of*

**Prof. SHEETAL VA**  
**Assistant Professor, BMSCE**

*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**  
**Sep-2020 to Jan-2021**

**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 RECORD carried out by **S SHREE LAKSHMI (IBM19CS136)** who is the bonafide students 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 Visveswaraiyah Technological University, Belgaum during the year 2020-2021. The lab report has been approved as it satisfies the academic requirements in respect of **DATA STRUCTURE LAB RECORD (19CS3PCDST)** work prescribed for the said degree.

Signature of the Guide  
Prof. Prof. Sheelal VA  
Assistant Professor  
BMSCE, Bengaluru

Signature of the HOD  
Dr. Umadevi V  
Associate Prof.& Head, Dept. of CSE  
BMSCE, Bengaluru

External Viva

Name of the Examiner

Signature with date

1. \_\_\_\_\_

\_\_\_\_\_

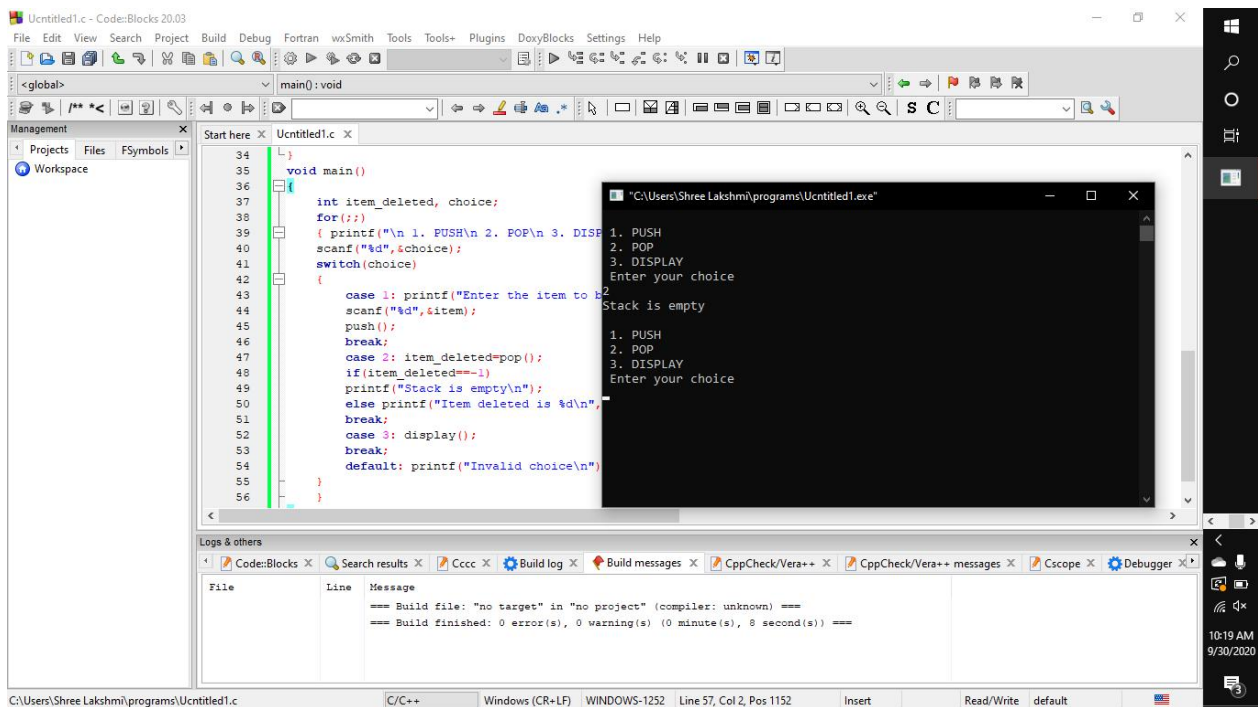
2. \_\_\_\_\_

\_\_\_\_\_

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



Code:

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

```
#include <conio.h>
```

```
#define STACK_SIZE 5
```

```
int top=-1,s[10],item;
```

```
void push()
```

```
{
```

```
    if(top==STACK_SIZE-1)
```

```
    { printf("Stack overflow\n");
```

```

    return; }

    top=top+1;
    s[top]=item;
}

int pop()
{
    if(top== -1) return -1;
    return s[top--];
}

void display()
{
    int i;
    if(top== -1)
    {
        printf("Stack is empty\n");
        return;
    }
    printf("Contents of the stack are:\n");
    for(i=0;i<=top;i++)
    {
        printf("%d\n",s[i]);

    }
}

void main()
{
    int item_deleted, choice;

    for(;;)
    { printf("\n 1. PUSH\n 2. POP\n 3. DISPLAY\n Enter your choice\n");

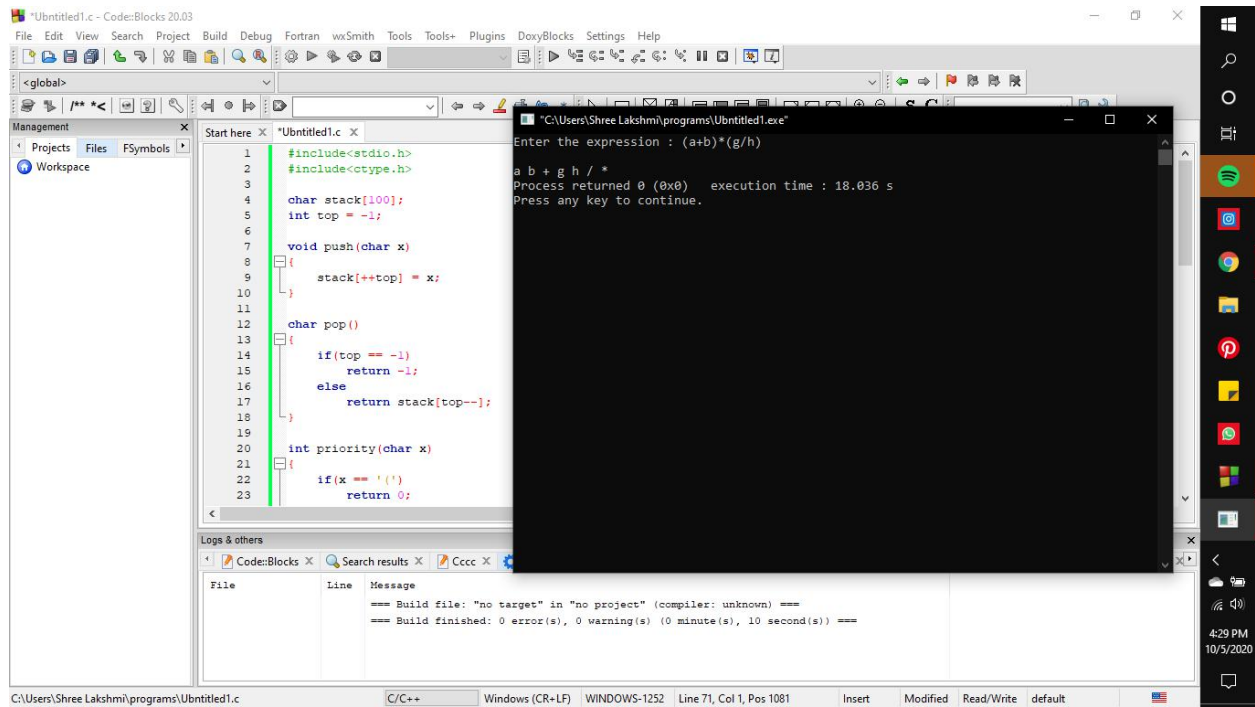
```

```

scanf("%d",&choice);
switch(choice)
{
    case 1: printf("Enter the item to be inserted\n");
            scanf("%d",&item);
            push();
            printf("Item has been inserted\n");
            break;
    case 2: item_deleted=pop();
            if(item_deleted==-1)
                printf("Stack is empty\n");
            else printf("Item deleted is %d\n",item_deleted);
            break;
    case 3: display();
            break;
    default: printf("Invalid choice\n");
}
}
}

```

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)



### Source code:

```
#include <stdio.h>
#include <string.h>
int F(char symbol)
{
    switch(symbol)
    {
        case '+':
        case '-': return 2;
        case '*':
        case '/': return 4;
        case '^':
        case '$': return 5;
        case '(': return 0;
        case '#': return -1;
        default : return 8;
    }
}
```

```

}

int G(char symbol)
{
    switch(symbol)
    {
        case '+':
            case '-': return 1;
            case '*':
            case '/': return 3;
            case '^':
            case '$': return 6;
            case '(': return 9;
            case ')': return 0;
            default : return 7;
        }
    }

void infix_postfix(char infix[],char postfix[])
{
    int top,i,j; char s[30];
    top=-1;
    s[++top]='#';
    j=0;
    for(i=0;i<strlen(infix);i++)
    {
        char symbol=infix[i];
        while(F(s[top])>G(symbol))
        {
            postfix[j]=s[top--];
            j++;
        }
    }
}

```

```

        }
        if(F(s[top])!=G(symbol))
            s[++top]=symbol;
        else
            top--;
    }
    while (s[top]!='#')
    {
        postfix [j++]=s[top--];
    }
    postfix[j]='\0';
}

void main()
{
    char infix[20];
    char postfix[20];
    printf("Enter the valid infix expression\n");
    scanf("%s",&infix);
    infix_postfix(infix, postfix);
    printf("the postfix expression is :");
    printf("%s\n", postfix);
}

```

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 .



```
"C:\Users\Shree Lakshmi\programs\queue.exe"
78
queue overflow
1:insert
2:delete
3:display
4:exit
enter the choice
3
contents of queue
34
345
67

1:insert
2:delete
3:display
4:exit
enter the choice
2
item deleted=34

1:insert
2:delete
3:display
4:exit
enter the choice
2
item deleted=345

1:insert
2:delete
3:display
4:exit
enter the choice
2
item deleted=67

1:insert
2:delete
3:display
4:exit
enter the choice
2
queue is empty
```

**Code :**

```
#include<stdio.h>

#include<stdlib.h>

#include<process.h>

#define que_size 3

int item,front=0,rear=-1,q[que_size],count=0;

void insertrear()
{
    if(count==que_size)
    {
        printf("queue overflow");
        return;
    }
    rear=(rear+1)%que_size;
    q[rear]=item;
    count++;
}
```

```

}
int deletefront()
{
    if(count==0) return -1;
    item = q[front];
    front=(front+1)%que_size;
    count=count-1;
    return item;
}
void displayq()
{
    int i,f;
    if(count==0)
    {
        printf("queue is empty");
        return;
    }
    f=front;
    printf("contents of queue \n");
    for(i=0;i<=count;i++)
    {
        printf("%d\n",q[f]);
        f=(f+1)%que_size;
    }
}
void main()
{
    int choice;
    for(;;)

```

```

{
    printf("\n1.Insert rear \n2.Delete front \n3.Display \n4.exit \n ");
    printf("Enter the choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:printf("Enter the item to be inserted :");
            scanf("%d",&item);
            insertrear();
            break;
        case 2:item=deletefront();
            if(item== -1)
                printf("queue is empty\n");
            else
                printf("item deleted is %d \n",item);
            break;
        case 3:displayq();
            break;
        default:exit(0);
    }
}

getch();
}

```

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



```
"C:\Users\Shree Lakshmi\programs\LB4.exe"
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 1
Enter the item to be inserted :45
queue overflow
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 3
contents of queue
1
2
3
1
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 2
item deleted is 1
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 2
item deleted is 2
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 2
item deleted is 3
1.Insert rear
2.Delete front
3.Display
4.exit
Enter the choice : 2
queue is empty
```

### Source code:

```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
#define que_size 3
int item,front=0,rear=-1,q[que_size],count=0;
void insertrear()
{
    if(count==que_size)
    {
        printf("queue overflow");
        return;
    }
    rear=(rear+1)%que_size;
    q[rear]=item;
```

```

        count++;
    }
int deletefront()
{
    if(count==0) return -1;
    item = q[front];
    front=(front+1)%que_size;
    count=count-1;
    return item;
}
void displayq()
{
    int i,f;
    if(count==0)
    {
        printf("queue is empty");
        return;
    }
    f=front;
    printf("contents of queue \n");
    for(i=0;i<=count;i++)
    {
        printf("%d\n",q[f]);
        f=(f+1)%que_size;
    }
}
void main()
{
    int choice;
    for(;;)

```

```

{
    printf("\n1.Insert rear \n2.Delete front \n3.Display \n4.exit \n ");
    printf("Enter the choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:printf("Enter the item to be inserted :");
                scanf("%d",&item);
                insertrear();
                break;
        case 2:item=deletefront();
                if(item==-1)
                    printf("queue is empty\n");
                else
                    printf("item deleted is %d \n",item);
                break;
        case 3:displayq();
                break;
        default:exit(0);
    }
}
getch();
}

```

5.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

```
"C:\Users\Shree Lakshmi\programs\LB5.exe"
6:Delete element from a given position
7:Display list
8:Exit
Enter the choice: 1
Enter the item at rear end
23

1:Insert rear
2:Delete rear
3:Insert front
4:Delete front
5:Insert element at a given position
6:Delete element from a given position
7:Display list
8:Exit
Enter the choice: 3
Enter the item at front end
34

1:Insert rear
2:Delete rear
3:Insert front
4:Delete front
5:Insert element at a given position
6:Delete element from a given position
7:Display list
8:Exit
Enter the choice: 5
Enter the item to be inserted at given position
45
Enter the position
2

1:Insert rear
2:Delete rear
3:Insert front
4:Delete front
5:Insert element at a given position
6:Delete element from a given position
7:Display list
8:Exit
Enter the choice: 7
34
45
23
```

### Code :

```
#include<stdio.h>

#include<stdlib.h>

struct node{

int info;

struct node *link;

};

typedef struct node *NODE;

NODE getnode(){

NODE x;

x=(NODE)malloc(sizeof(struct node));

if(x==NULL){

printf("Memory full\n");

exit(0);

}

return x;

}
```

```

void freenode(NODE x){
    free(x);
}

NODE insert_front(NODE first,int item){
    NODE temp;
    temp=getnode();
    temp->info=item;
    temp->link=NULL;
    if(first==NULL)
        return temp;
    temp->link=first;
    first=temp;
    return first;
}

NODE delete_front(NODE first){
    NODE temp;
    if(first==NULL){
        printf("List is empty cannot delete\n");
        return first;
    }
    temp=first;
    temp=temp->link;
    printf("Item deleted at front end is %d\n",first->info);
    free(first);
    return temp;
}

NODE insert_rear(NODE first,int item){
    NODE temp,cur;
    temp=getnode();
    temp->info=item;

```



```

temp->link=NULL;
if(first==NULL)
return temp;
cur=first;
while(cur->link!=NULL)
cur=cur->link;
cur->link=temp;
return first;
}
NODE delete_rear(NODE first){
NODE cur,prev;
if(first==NULL){
printf("List is empty cannot delete\n");
return first;
}
if(first->link==NULL){
printf("Item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL){
prev=cur;
cur=cur->link;
}
printf("Item deleted at rear end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;

```

```

}
NODE insert_pos(int item,int pos,NODE first){
NODE temp,cur,prev;
int count;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL&&pos==1){
return temp;
}
if(first==NULL){
printf("Invalid position\n");
return first;
}
if(pos==1){
temp->link=first;
first=temp;
return temp;
}
count=1;
prev=NULL;
cur=first;
while(cur!=NULL&&count!=pos){
prev=cur;
cur=cur->link;
count++;
}
if(
count==pos){
prev->link=temp;

```

```

temp->link=cur;
return first;
}
printf("Invalid position\n");
return first;
}
NODE delete_pos(int pos,NODE first){
NODE cur;
NODE prev;
int count,flag=0;
if(first==NULL || pos<0){
printf("Invalid position\n");
return NULL;
}
if(pos==1){
cur=first;
first=first->link;
freenode(cur);
return first;
}
prev=NULL;
cur=first;
count=1;
while(cur!=NULL){
if(count==pos){
flag=1;
break;
}
count++;
prev=cur;

```

```

cur=cur->link;
}
if(flag==0){
printf("Invalid position\n");
return first;
}
printf("Item deleted at given position is %d\n",cur->info);
prev->link=cur->link;
freenode(cur);
return first;
}
void display(NODE first){
NODE temp;
if(first==NULL)
printf("List empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link){
printf("%d\n",temp->info);
}
}

void main()
{
int item,choice,key,pos;
int count=0;
NODE first=NULL;
for(;;){
printf("\n1:Insert rear\n2:Delete rear\n3:Insert front\n4:Delete front\n5:Insert element at a given position\n6:Delete element from a given position\n7:Display list\n8:Exit\n");
printf("Enter the choice: ");
scanf("%d",&choice);

```

```

switch(choice){
case 1:printf("Enter the item at rear end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 2:first=delete_rear(first);
break;
case 3:printf("\nEnter the item at front end\n");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 4:first=delete_front(first);
break;
case 5:printf("Enter the item to be inserted at given position\n");
scanf("%d",&item);
printf("Enter the position\n");
scanf("%d",&pos);
first=insert_pos(item,pos,first);
break;
case 6:printf("Enter the position\n");
scanf("%d",&pos);
first=delete_pos(pos,first);
break;
case 7:display(first);
break;

default:exit(0);
break;
}
}

```

}

6.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.

```
"C:\Users\Shree Lakshmi\programs\LB5.exe"
1:Insert rear
2:Delete rear
3:Insert front
4:Delete front
5:Insert element at a given position
6:Delete element from a given position
7:Display list
8:Exit
Enter the choice: 4
Item deleted at front end is 34

1:Insert rear
2:Delete rear
3:Insert front
4:Delete front
5:Insert element at a given position
6:Delete element from a given position
7:Display list
8:Exit
Enter the choice: 6
Enter the position
2
Invalid position

1:Insert rear
2:Delete rear
3:Insert front
4:Delete front
5:Insert element at a given position
6:Delete element from a given position
7:Display list
8:Exit
Enter the choice: 6
Enter the position
1

1:Insert rear
2:Delete rear
3:Insert front
4:Delete front
5:Insert element at a given position
6:Delete element from a given position
7:Display list
8:Exit
Enter the choice: 7
List empty cannot display items
```

**Code :**

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

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

```
struct node{
```

```
int info;
```

```
struct node *link;
```

```
};
```

```
typedef struct node *NODE;
```

```
NODE getnode(){
```

```
NODE x;
```

```

x=(NODE)malloc(sizeof(struct node));

if(x==NULL){
printf("Memory full\n");
exit(0);
}

return x;
}

void freenode(NODE x){
free(x);
}

NODE insert_front(NODE first,int item){
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}

NODE delete_front(NODE first){
NODE temp;
if(first==NULL){
printf("List is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("Item deleted at front end is %d\n",first->info);

```

```

free(first);
return temp;
}
NODE insert_rear(NODE first,int item){
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
cur=first;
while(cur->link!=NULL)
cur=cur->link;
cur->link=temp;
return first;
}
NODE delete_rear(NODE first){
NODE cur,prev;
if(first==NULL){
printf("List is empty cannot delete\n");
return first;
}
if(first->link==NULL){
printf("Item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL){

```



```

prev=cur;
cur=cur->link;
}
printf("Item deleted at rear end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}

NODE insert_pos(int item,int pos,NODE first){
NODE temp,cur,prev;
int count;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL&&pos==1){
return temp;
}
if(first==NULL){
printf("Invalid position\n");
return first;
}
if(pos==1){
temp->link=first;
first=temp;
return temp;
}
count=1;
prev=NULL;
cur=first;
while(cur!=NULL&&count!=pos){

```

```

prev=cur;
cur=cur->link;
count++;
}
if(
count==pos){
prev->link=temp;
temp->link=cur;
return first;
}
printf("Invalid position\n");
return first;
}
NODE delete_pos(int pos,NODE first){
NODE cur;
NODE prev;
int count,flag=0;
if(first==NULL || pos<0){
printf("Invalid position\n");
return NULL;
}
if(pos==1){
cur=first;
first=first->link;
freenode(cur);
return first;
}
prev=NULL;
cur=first;
count=1;

```

```

while(cur!=NULL){
if(count==pos){
flag=1;
break;
}
count++;
prev=cur;
cur=cur->link;
}
if(flag==0){
printf("Invalid position\n");
return first;
}
printf("Item deleted at given position is %d\n",cur->info);
prev->link=cur->link;
freenode(cur);
return first;
}

void display(NODE first){
NODE temp;
if(first==NULL)
printf("List empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link){
printf("%d\n",temp->info);
}
}

void main()
{
int item,choice,key,pos;

```

```

int count=0;
NODE first=NULL;
for(;;){
printf("\n1:Insert rear\n2:Delete rear\n3:Insert front\n4:Delete front\n5:Insert element at a given
position\n6:Delete element from a given position\n7:Display list\n8:Exit\n");

printf("Enter the choice: ");
scanf("%d",&choice);
switch(choice){
case 1:printf("Enter the item at rear end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 2:first=delete_rear(first);
break;
case 3:printf("\nEnter the item at front end\n");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 4:first=delete_front(first);
break;
case 5:printf("Enter the item to be inserted at given position\n");
scanf("%d",&item);
printf("Enter the position\n");
scanf("%d",&pos);
first=insert_pos(item,pos,first);
break;
case 6:printf("Enter the position\n");
scanf("%d",&pos);
first=delete_pos(pos,first);
break;

```

```
case 7:display(first);
```

```
break;
```

```
default:exit(0);
```

```
break;
```

```
}
```

```
}
```

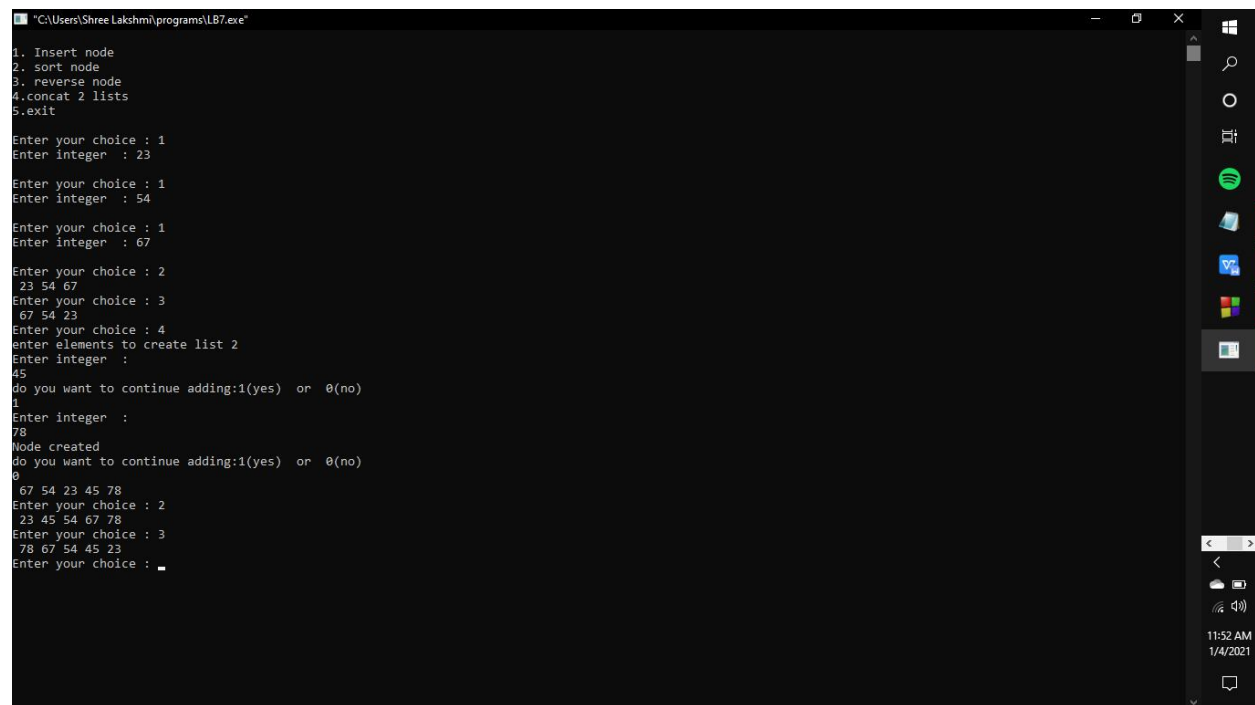
```
}
```

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



```
"C:\Users\Shree Lakshmi\programs\LB7.exe"
1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit
Enter your choice : 1
Enter integer : 23
Enter your choice : 1
Enter integer : 54
Enter your choice : 1
Enter integer : 67
Enter your choice : 2
23 54 67
Enter your choice : 3
67 54 23
Enter your choice : 4
enter elements to create list 2
Enter integer :
45
do you want to continue adding:1(yes) or 0(no)
1
Enter integer :
78
Node created
do you want to continue adding:1(yes) or 0(no)
0
67 54 23 45 78
Enter your choice : 2
23 45 54 67 78
Enter your choice : 3
78 67 54 45 23
Enter your choice : 5
```

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

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

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

```

struct node
{
    int sem;
    struct node *next;
};

struct node *head= NULL;
struct node *head2= NULL;
int c=0;
void Insert()
{
    struct node *newnode;
    struct node *temp;
    int s;
    printf("Enter integer : ");
    scanf("%d",&s);
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->sem =s;
    if (head==NULL)
    {
        newnode->next=NULL;
        head=newnode;

        c++;
    }
    else
    {
        temp=head;
        while(temp->next!=NULL)
        {
            temp=temp->next;

```

```

    }

    temp->next=newnode;
    newnode->next=NULL;
    c++;
    \
    }
}
void Insert2()
{
    struct node *newnode;
    struct node *temp;

    int s,y;
    printf("enter elements to create list 2\n");
    do
    {
        printf("Enter integer : \n");
        scanf("%d",&s);
        newnode=(struct node*)malloc(sizeof(struct node));
        newnode->sem =s;
        if (head2==NULL)
        {
            newnode->next=NULL;
            head2=newnode;
            c++;
        }
        else
        {
            temp=head2;
            while(temp->next!=NULL)
            {

```

```

        temp=temp->next;
    }

    temp->next=newnode;
    newnode->next=NULL;
    c++;
    printf("Node created\n");
}
printf("do you want to continue adding:1(yes) or 0(no)\n");
scanf("%d",&y);
}while(y!=0);
}

```

```

void bubbleSort()
{
    int swapped, i;
    struct node *ptr1;
    struct node *lptr = NULL;

```

```

    if (head == NULL)
        return;

```

```

do

```

```

{
    swapped = 0;
    ptr1 = head;

```

```

    while (ptr1->next != lptr)
    {

```



```

        if (ptr1->sem > ptr1->next->sem)
        {
            int temp = ptr1->sem;
            ptr1->sem = ptr1->next->sem;
            ptr1->next->sem = temp;
            swapped = 1;
        }
        ptr1 = ptr1->next;
    }
    lptr = ptr1;
}
while (swapped);
}

```

```

void reverse()
{
    struct node* prev = NULL;
    struct node* current = head;
    struct node* next ;
    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    head= prev;
}

```

```

void concat()
{

```

```

    struct node *ptr;
    if(head==NULL)
    {
        head=head2;
    }
    if(head2==NULL)
    {
        head2=head;
    }
    ptr=head;
    while(ptr->next!=NULL)
        ptr=ptr->next;
    ptr->next=head2;
}

void display1()
{
    struct node *ptr;
    ptr=head;
    int i=1;

    if(ptr==NULL)
    {
        printf("Linked list is empty!\n");
    }
    else
    {
        while(ptr!= NULL)
        {
            printf(" %d",ptr->sem);
            i++;

```

```

        ptr=ptr->next;

    }

}

}
void display2()
{
    struct node *ptr;
    ptr=head2;
    int i=1;

    if(ptr==NULL)
    {
        printf("Linked list is empty!\n");
    }
    else
    {
        while(ptr!= NULL)
        {

            printf(" %d",ptr->sem);
            printf("\n");
            i++;
            ptr=ptr->next;

        }

    }
}

```

```
}
```

```
int main()
```

```
{    printf("\n1. Insert node \n2. sort node\n3. reverse node\n4.concat 2 lists \n5.exit\n");
```

```
    int choice,pos;
```

```
    do
```

```
    {
```

```
        printf("\nEnter your choice : ");
```

```
        scanf("%d",&choice);
```

```
        switch(choice)
```

```
        {
```

```
            case 1:
```

```
                Insert();
```

```
                break;
```

```
            case 2:
```

```
                bubbleSort();
```

```
                display1();
```

```
                break;
```

```
            case 3:
```

```
                reverse();
```

```
                display1();
```

```
                break;
```

```
            case 4:
```

```
                Insert2();
```

```
                concat();
```

```

        display1();
        break;

    case 5:
        break;

    default:
        printf("Wrong choice!\n");
        break;
    }
} while(choice!=5);
return 0;
}

```

## 8. WAP to implement Stack & Queues using Linked Representation

```

"C:\Users\Shree Lakshmi\programs\LB8.exe"
1.Insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?1
Enter value?
24

1.Insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?1
Enter value?
25

1.Insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?3
printing values .....
21
22
23
24
25

1.Insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?2

```

```
"C:\Users\Shree Lakshmi\programs\LB8.exe"
4.Exit
Enter your choice ?2
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?2
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?2
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?2
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?2
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
Enter your choice ?2
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
UNDERFLOW
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
```

```
#include<stdio.h>

#include<stdlib.h>

struct node
{
    int data;
    struct node *next;
};

struct node *front;
struct node *rear;

void insert();
void deleteq();
void display();

int main ()
{
    int choice;
    while(choice != 4)
    {
```

```

printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");
printf("\nEnter your choice ?");
scanf("%d",& choice);
switch(choice)
{
    case 1:
        insert();
        break;
    case 2:
        deleteq();
        break;
    case 3:
        display();
        break;
    case 4:
        exit(0);
        break;
    default:
        printf("\nEnter valid choice??\n");
}
}
}

void insert()
{
    struct node *ptr;
    int item;

    ptr = (struct node *) malloc (sizeof(struct node));

```

```

if(ptr == NULL)
{
    printf("\nOVERFLOW\n");
    return;
}
else
{
    printf("\nEnter value?\n");
    scanf("%d",&item);
    ptr -> data = item;
    if(front == NULL)
    {
        front = ptr;
        rear = ptr;
        front -> next = NULL;
        rear -> next = NULL;
    }
    else
    {
        rear -> next = ptr;
        rear = ptr;
        rear->next = NULL;
    }
}

void deleteq()
{
    struct node *ptr;
    if(front == NULL)
    {

```



```

        printf("\nUNDERFLOW\n");
        return;
    }
    else
    {
        ptr = front;
        front = front -> next;
        free(ptr);
    }
}

void display()
{
    struct node *ptr;
    ptr = front;
    if(front == NULL)
    {
        printf("\nEmpty queue\n");
    }
    else
    {
        printf("\nprinting values ..... \n");
        while(ptr != NULL)
        {
            printf("\n%d\n", ptr -> data);
            ptr = ptr -> next;
        }
    }
}

```

## 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

```
"C:\Users\Shree Lakshmi\programs\LB9.exe"
1.Insert to the left
2.Delete
3.Display
enter your choice : 1
enter value to be inserted 12
enter your choice : 1
enter value to be inserted 22
enter your choice : 1
enter value to be inserted 32
enter your choice : 1
enter value to be inserted 42
enter your choice : 1
enter value to be inserted 52
enter your choice : 3
52 42 32 22 12
enter your choice : 2
Enter the value22
Node Deleted
enter your choice : 2
Enter the value42
Node Deleted
enter your choice :
```

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

```
#include<conio.h>
```

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

```
struct node{
```

```
    int data;
```

```
    struct node *next;
```

```
    struct node *prev;
```

```
};
```

```
struct node *head;
```

```
void insertbeginning(int item)
```

```
{    struct node *ptr  = (struct node *)malloc(sizeof(struct node));
```

```
    if(head==NULL)
```

```

    { ptr->next = NULL;
      ptr->prev=NULL;
      ptr->data=item;
      head=ptr;  }
else
{   ptr->data=item;
    ptr->prev=NULL;
    ptr->next = head;
    head->prev=ptr;
    head=ptr;  }

}

```

```

void delete_specified( )
{   struct node *ptr, *temp;
    int val;
    printf("Enter the  value");
    scanf("%d",&val);
    temp = head;
    while(temp -> data != val)
    temp = temp -> next;
    if(temp -> next == NULL)
    {   printf("\nCan't delete\n");   }
    else if(temp -> next -> next == NULL)
    {   temp ->next = NULL;
        printf("\nNode Deleted\n");

    }
    else

```

```

    { ptr = temp -> next;
      temp -> next = ptr -> next;
      ptr -> next -> prev = temp;
      free(ptr);
      printf("\nNode Deleted\n");

    }
}

```

```

void display()
{
    struct node *ptr;
    ptr=head;
    if(ptr==NULL)
    {
        printf("empty ");
    }
    else
    {
        while(ptr!=NULL)
        {
            printf("%d ",ptr->data);
            ptr=ptr -> next;
        }
        printf("\n");
    }
}

```

```

void main()
{

```

```

int op=0;int a;
printf("\n 1.Insert to the left\n2.Delete\n3.Display\n ");

while(op!=4)
{
    printf("\nEnter your choice : ");
    scanf("%d",&op);
    switch(op)
    {
        case 1:printf("Enter value to be inserted ");
                scanf("%d",&a);
                insertbeginning(a);
                break;
        case 2:delete_specified();
                break;
        case 3: display();
                break;

    }
}
}

```

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 post order
- c) To display the elements in the tree.

```
"C:\Users\Shree Lakshmi\programs\LB10.exe"
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
1
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
2
give direction to insert
1
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
3
give direction to insert
r
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
4
give direction to insert
l
1.insert
2.preorder
3.inorder
4.postorder
```

```
"C:\Users\Shree Lakshmi\programs\LB10.exe"
enter the choice
1
enter the item
45
give direction to insert
lr
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
6
give direction to insert
rl
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
7
give direction to insert
rr
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
7
3
6
1
45
2
4
1.insert
2.preorder
3.inorder
4.postorder
```

```
"C:\Users\Shree Lakshmi\programs\LB10.exe"
enter the choice
2
given tree is
    7
   / \
  3   6
 / \
1   45
 / \
2   4

the preorder traversal is
the item is 1
the item is 2
the item is 4
the item is 45
the item is 3
the item is 6
the item is 7
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
3
given tree is
    7
   / \
  3   6
 / \
1   45
 / \
2   4

the inorder traversal is
the item is 4
the item is 2
the item is 45
the item is 1
the item is 6
the item is 3
the item is 7
1.insert
2.preorder
3.inorder
4.postorder
5.display
```

```
"C:\Users\Shree Lakshmi\programs\LB10.exe"
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
4
given tree is
    7
   / \
  3   6
 / \
1   45
 / \
2   4

the postorder traversal is
the item is 4
the item is 45
the item is 2
the item is 6
the item is 7
the item is 3
the item is 1
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
    7
   / \
  3   6
 / \
1   45
 / \
2   4
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
```

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<string.h>

struct node
```

```

{
int info;
struct node*llink;
struct node*rlink;
};
typedef struct node*NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("memory not available");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert(int item,NODE root)
{
NODE temp,cur,prev;
char direction[10];
int i;
temp=getnode();
temp->info=item;
temp->llink=NULL;
temp->rlink=NULL;

```



```

if(root==NULL)
    return temp;
printf("give direction to insert\n");
scanf("%s",direction);
prev=NULL;
cur=root;
for(i=0;i<strlen(direction)&&cur!=NULL;i++)
{
    prev=cur;
    if(direction[i]=='l')
        cur=cur->llink;
    else
        cur=cur->rlink;
}
if(cur!=NULL||i!=strlen(direction))
{
    printf("insertion not possible\n");
    freenode(temp);
    return(root);
}
if(cur==NULL)
{
    if(direction[i-1]=='l')
        prev->llink=temp;
    else
        prev->rlink=temp;
}
return(root);
}
void preorder(NODE root)

```

```

{
if(root!=NULL)
{
printf("the item is %d\n",root->info);
preorder(root->llink);
preorder(root->rlink);
}
}

void inorder(NODE root)
{
if(root!=NULL)
{
inorder(root->llink);
printf("the item is %d\n",root->info);
inorder(root->rlink);
}
}

void postorder(NODE root)
{
if (root!=NULL)
{
postorder(root->llink);
postorder(root->rlink);
printf("the item is %d\n",root->info);
}
}

void display(NODE root,int i)
{
int j;
if(root!=NULL)

```

```

{
display(root->rlink,i+1);
for (j=1;j<=i;j++)
printf(" ");
printf("%d\n",root->info);
display(root->llink,i+1);
}
}

void main()
{
NODE root=NULL;
int choice,i,item;
for(;;)
{
printf("1.insert\n2.preorder\n3.inorder\n4.postorder\n5.display\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("enter the item\n");
scanf("%d",&item);
root=insert(item,root);
break;
case 2: if(root==NULL)
{
printf("tree is empty");
}
else
{

```

```

        printf("given tree is\n");
        display(root,1);
        printf("the preorder traversal is \n");
        preorder(root);
    }
    break;
case 3:if(root==NULL)
    {
        printf("tree is empty\n");
    }
    else
    {
        printf("given tree is\n");
        display(root,1);
        printf("the inorder traversal is \n");
        inorder(root);
    }
    break;
case 4:if (root==NULL)
    {
        printf("tree is empty");
    }
    else
    {
        printf("given tree is\n");
        display(root,1);
        printf("the postorder traversal is \n");
        postorder(root);
    }
    break;

```

```
case 5:display(root,1);
```

```
    break;
```

```
default:exit(0);
```

```
}
```

```
}
```

```
}
```