VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

DATA STRUCTURES

Submitted by

SNEHAL BANDI (1BM21CS214)

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 **SNEHAL BANDI (1BM21CS214),** 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.

Name of the Lab-Incharge **Dr. Jyothi S Nayak** Designation Professor and Head Department
of CSE Department of CSE
BMSCE, Bengaluru BMSCE, Bengaluru
`

Index Sheet

Course Outcome

CO1	Apply the concept of linear and nonlinear data structure.
CO2	Analyse 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.

```
#include<stdio.h>

#include<conio.h>

#define SIZE 3

int STACK[SIZE],TOP=-1,ITEM;

void push();

void pop();

void display();


void main()

{ int choice;

 while(1)

 {

 printf(‘‘\n\n 1:push\n 2:pop\n 3:display\n 4:exit\n’‘);

 printf(‘‘enter your choice’‘);

 scanf(‘‘%d’‘,&choice);

 switch(choice)

 {

 case 1:push();

 break;

 case 2: pop();

 break;

 case 3: display();

 break;

 case 4: exit(0);
```



```
 break;
```

```
 default: printf(‘‘wrong choice’‘);


 }
 }
 getch();
}
void push()
{
 if(TOP==SIZE-1)
 {
 printf(‘‘stack overflow’‘);
 return;
 }
 else
 {
 printf(‘‘enter an element\n’‘);
 scanf(‘‘%d’‘,&ITEM);
 printf(‘‘entered element is %d\n\n’‘,ITEM);
TOP=TOP+1;
 STACK[TOP]=ITEM;
 }
}
void pop()
{
 int del;
 if(TOP==-1)
```

```
 {
 printf(‘‘stack underflow\n’‘);
return;
 }
 else
 {
 del=STACK[TOP];
 printf(‘‘poped element is %d\n’‘,del);
TOP=TOP-1;
 }
}
void display()
{
 int i;
if(TOP==-1)
{
 printf(‘‘STACK IS EMPTY\n’‘);
 return;
}
else
{
 for(i=TOP;i>=0;i--)
 {
 printf(‘‘%d\n’‘,STACK[i]);
 }
}
}
```

```
22
entered element is 22



 1:push
 2:pop
 3:display
 4:exit
enter your choice1
enter an element
44
entered element is 44



 1:push
 2:pop
 3:display
 4:exit
enter your choice1
enter an element
55
entered element is 55



 1:push
 2:pop
 3:display
 4:exit
enter your choice1
stack overflow
```

```
 1:push
 2:pop
 3:display
 4:exit
enter your choice3
44
22



 1:push
 2:pop
 3:display
 4:exit
enter your choice2
poped element is 44


 1:push
 2:pop
 3:display
 4:exit
enter your choice2
poped element is 22


 1:push
 2:pop
 3:display
 4:exit
enter your choice2
stack underflow


 1:push
 2:pop
 3:display
 4:exit
```

```
 1:push
 2:pop
 3:display
 4:exit
enter your choice1
enter an element
22
entered element is 22



 1:push
 2:pop
 3:display
 4:exit
enter your choice1
enter an element
44
entered element is 44



 1:push
 2:pop
 3:display
 4:exit
enter your choice1
enter an element
55
entered element is 55



 1:push
 2:pop
 3:display
 4:exit
enter your choice
```

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).

```
#include<stdio.h>
#include<string.h>
int index=0,pos=0,top=-1,length;
char symbol,temp,infix[20],postfix[20],stack[20];
void infix_postfix();
void push(char symbol);
char pop();
int pred(char symbol);
void main(){
printf("enter infix expression");
scanf("%s", infix);
infix_postfix();
printf("infix expression=%s",infix);
printf("postfix expression=%s",postfix);
getch();
}

void infix_postfix()
{

 length=strlen(infix);
 push('#');
 while(index<length)
 {
```

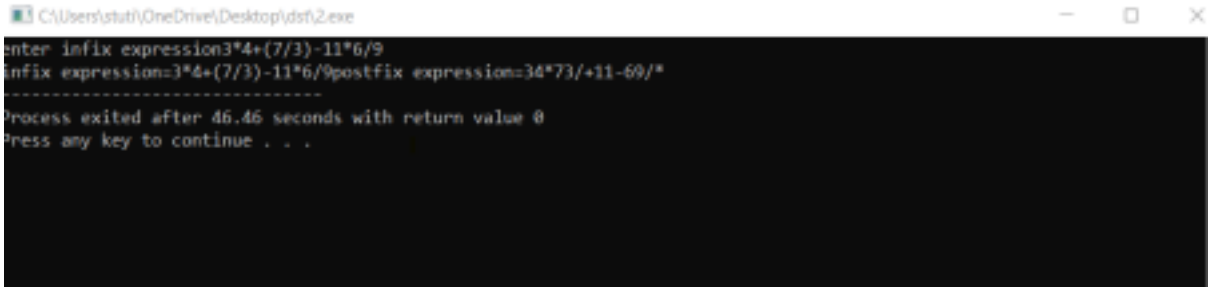
```
symbol = infix[index];
switch(symbol)
{
case '(' : push(symbol);
break;
case ')': temp=pop();
while(temp!='(')
{
postfix[pos]=temp;
pos++;
temp=pop();
}
break;
case'+':
case'-':
case'*':
case'/':
case'^':while(pred(stack[top])>=pred(symbol))  {
temp=pop();
postfix[pos++]=temp;
}
push(symbol);
break;
default: postfix[pos++]=symbol;
}
```

```
 index++;
 }
 while(top>0)
 {
 temp=pop();
 postfix[pos++]=temp;  }
 }

 void push(char symbol)
{

 top=top+1;
 stack[top]=symbol;  }

 char pop()
 {

 char symb;
 symb=stack[top];
top=top-1;
 return(symb);
 }

 int pred(char symbol)
{
```



```
 int p;
```

```
switch(symbol)
{
case'^': p=3;
break;
case'/': p=2;
break;
case'*':
case'+':
case'-':p=1;
break;
case'(': p=0;
break;
case'#': p=-1;
break;
}
return(p);
}
```



LAB PROGRAM3: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.

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#define SIZE 3

int queue[SIZE],rear=-1,front=0,ITEM;

void push();

void pop();

void display();


void main()

{ int choice;

while(1)

{

printf("\n\n 1:push\n 2:pop\n 3:display\n 4:exit\n");

printf("enter your choice");

scanf("%d",&choice);

switch(choice)

{

case 1:push();

break;

case 2: pop();

break;

case 3: display();

break;
```

```
case 4: exit(0);
break;
default: printf("wrong choice");

}
}
getch();
}
void push()
{

if(rear==SIZE-1)
{
printf("queue is full");
}
else
{
printf("enter an element\n");
scanf("%d",&ITEM);
printf("entered element is
%d\n\n",ITEM); rear++;
queue[rear]=ITEM;
}
}
void pop()
{
```

```
int del;
```



```
if(rear==-1)
printf("queue is
empty\n"); else
{
del=queue[front];
front++;
if(front==SIZE)
{
front=0;
rear=-1;

}
}
}

void display()
{
int i;
if(rear==-1)
{
printf("QUEUE IS
EMPTY\n"); }
else
{
for(i=front;i<=rear;i++)
```

```
{
printf("%d\n",queue[i]);
}
}
}
```



```
enter an element
69
entered element is 69


 1:push
 2:pop
 3:display
 4:exit
enter your choice1
enter an element
71
entered element is 71


 1:push
 2:pop
 3:display
 4:exit
enter your choice1
queue is full

 1:push
 2:pop
 3:display
 4:exit
enter your choice3
89
69
71


 1:push
 2:pop
 3:display
 4:exit
enter your choice
```

```
 1:push
 2:pop
 3:display
 4:exit
enter your choice2
queue is empty


 1:push
 2:pop
 3:display
 4:exit
enter your choice1
enter an element
89
entered element is 89



 1:push
 2:pop
 3:display
 4:exit
enter your choice1
enter an element
69
entered element is 69



 1:push
 2:pop
 3:display
 4:exit
enter your choice
```

e

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.

```
#include<stdio.h>
#include<stdlib.h>
#define max 6
int queue[max];
int front=-1;
int rear=-1;

void enqueue(int element)
{

 if(front==-1 && rear == -1)
 {
 front=0;
 rear=0;
 queue[rear]=element;
 }
 else if((rear+1)%max==front)
 {
 printf("queue is overflow");
 }
 else{
 rear=(rear+1)%max;
 queue[rear]=element;
 }
}

int dequeue()
{

 if((front==-1)&&(rear==-1))
 {

 printf("\n queue is underflow");
```

```
 }
 else if(front==rear)
 {
 printf("\n the dequeued element is %d", queue[front]);
front=-1;
 rear=-1;
 }
 else{
 printf("\n the dequeued element is %d", queue[front]);
front=(front+1)%max;
 }
}

void display()
{

 int i=front;
 if(front==-1 && rear==-1)
 {
 printf("\n queue is empty");
 }
 else
 {
 printf("\n elements in a queue are:");
 while(i<=rear)
 {
 printf("%d\n", queue[i]);
 i=(i+1)%max;
 }
 }
}

int main()
{

 int choice=1,x;

 while(1)
```

```
{
printf("\n 1. insert an element\n");
printf("\n 2. delete an element\n");
printf("\n 3. display all elements\n");
printf("\n 4. exit \n");
printf("\n enter your choice");
scanf("%d", &choice);

switch(choice)
{

case 1 : printf("\n enter element to be inserted\n");
scanf("%d",&x);
enqueue(x);
break;

case 2 : dequeue();
break;

case 3: display();
 break;
case 4: exit(0);
break;
default : printf("enter a valid choice");  }
}
return(0);
}
```

```
1. insert an element

2. delete an element

3. display all elements

4. exit

enter your choice2

the dequeued element is 23
1. insert an element

2. delete an element

3. display all elements

4. exit

enter your choice2

the dequeued element is 55
1. insert an element

2. delete an element

3. display all elements

4. exit

enter your choice2

queue is underflow
1. insert an element
```

```
1. insert an element

2. delete an element

3. display all elements

4. exit

enter your choice1

enter element to be inserted
5

1. insert an element

2. delete an element

3. display all elements

4. exit

enter your choice3

elements in a queue are:23
```

```
1. insert an element

2. delete an element

3. display all elements

4. exit

enter your choice2

queue is underflow
1. insert an element

2. delete an element

3. display all elements

4. exit

enter your choice3

queue is empty
1. insert an element

2. delete an element

3. display all elements

4. exit

enter your choice1

enter element to be inserted
23
```

LAB PROGRAM 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.

```
#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>


void create();

void display();

void insert_head();

void insert_last();

void insert_val();


struct Node

{

 int data;

 struct Node *link;

};

typedef struct Node node;

 node *start=NULL;


int main()

{

 int ch;

 while(1)

 {

 printf("\n1.Create 2.Display 3.Insert Head 4.Insert Last 5.Insert val 6.Exit");

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

```
scanf("%d",&ch);
switch(ch)
{
case 1:
create();
break;
case 2:
display();
break;
case 3:
insert_head();
break;
case 4:
insert_last();
break;
case 5: insert_val();
break;
case 6:
exit(1);

default:
printf("Invalid choice\n");  }
}
return 0;
}
```

```
void create()
{
 int c;
 node *neww,*curr;
 start=(node *) malloc(sizeof(node));
 curr=start;
 printf("Enter element\n");
 scanf("%d",&start->data);
 while(1)
 {
 printf("Do you want to add another element(1/0)\n");
scanf("%d",&c);
 if(c==1)
 {
 neww=(node *) malloc(sizeof(node));
printf("Enter element:\n");
 scanf("%d",&neww->data);
 curr->link = neww;
 curr=neww;
 }
 else
 {
 curr->link=NULL;
 break;
 }
 }
}
void display()
```

```
{
 node *temp;
 if(start==NULL)
 {
 printf("Linked list is empty\n");
return;
 }
 temp=start;
 while(temp!=NULL)
 {
 printf("%d\t",temp->data);
temp = temp->link;
 }
}
void insert_head(){
 node *temp,*mew;
 mew = (node *) malloc(sizeof(node));
temp = start;
 printf("enter element value:\n");
scanf("%d",&mew->data);
 mew->link = start;
 start = mew;
}
void insert_last(){
 node *neww,*temp;
 neww = (node *) malloc(sizeof(node));
```

```
temp = start;
printf("enter element value:\n");
scanf("%d",&neww->data);
while(temp->link!=NULL)
        {
temp = temp->link;
}
temp->link = neww;
neww->link = NULL;
}
void insert_val(){
        int pos;
node *neww, *temp;
neww =(node*)malloc(sizeof(node));
printf("Enter element:\n");
scanf("%d",&neww->data);
printf("Enter position:\n");
scanf("%d",&pos);
if(pos==1)
{
neww->link=start;
start=neww;
return;
                }
                int i=1;
                temp=start;
```

```
while(i<(pos-1) && temp!=NULL)
{ temp=temp->link; i++; }

if(i==(pos-1))

{

        neww->link=temp->link;

        temp->link=neww;

        return;

}

if(temp==NULL)

{ printf("Invalid position!!\n");} }
```

```
1.Create 2.Display 3.Insert Head 4.Insert Last 5.Insert val 6.Exit
Enter your choice:
1
Enter element
10
Do you want to add another element(1/0)
0

1.Create 2.Display 3.Insert Head 4.Insert Last 5.Insert val 6.Exit
Enter your choice:
3
enter element value:
20

1.Create 2.Display 3.Insert Head 4.Insert Last 5.Insert val 6.Exit
Enter your choice:
3
enter element value:
30

1.Create 2.Display 3.Insert Head 4.Insert Last 5.Insert val 6.Exit
Enter your choice:
4
enter element value:
40

1.Create 2.Display 3.Insert Head 4.Insert Last 5.Insert val 6.Exit
Enter your choice:
2
30      20      10      40
1.Create 2.Display 3.Insert Head 4.Insert Last 5.Insert val 6.Exit
Enter your choice:
5
Enter element:
50
Enter position:
3

1.Create 2.Display 3.Insert Head 4.Insert Last 5.Insert val 6.Exit
Enter your choice:
2
30      20      50      10      40
1.Create 2.Display 3.Insert Head 4.Insert Last 5.Insert val 6.Exit
```

LAB PROGRAM 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.

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
void create();
void display();
void delete_head();
void delete_last();
void delete_val();
struct Node
{
 int data;
 struct Node *link;
};
typedef struct Node node;
 node *start=NULL;
int main()
{
 int ch;
 while(1)
 {
 printf("1.Create 2.Display 3.Delete Head 4.Delete Last 5.Delete val 6.Exit\n");
printf("Enter your choice:\n");
 scanf("%d",&ch);
 switch(ch)
 {
```

```
 case 1:
 create();
 break;
 case 2:
 display();
 break;
 case 3: delete_head();
 break;
 case 4:delete_last();
 break;
 case 5:delete_val();
 break;
 case 6:
 exit(1);

 default:
 printf("Invalid choice\n");  }
 }
 return 0;
}
void create()
{
 int c;
 node *neww,*curr;
 start=(node *) malloc(sizeof(node));
```

```
curr=start;


 printf("Enter element: ");
 scanf("%d",&start->data);
 while(1)
 {
 printf("Do you want to add another element(1/0): ");
scanf("%d",&c);
 if(c==1)
 {
 neww=(node *) malloc(sizeof(node));
printf("Enter element: ");
 scanf("%d",&neww->data);
 curr->link = neww;
 curr=neww;
 }
 else
 {
 curr->link=NULL;
 break;
 }
 }
}
void display()
{
 node *temp;
```

```
 if(start==NULL)

 {

 printf("Linked list is empty\n");
 return;

 }

 temp=start;

 while(temp!=NULL)

 {

 printf("%d\t",temp->data);

temp = temp->link;  }

printf("\n");

}

void delete_head(){

 node *ptr;

 ptr = start;

 start=start->link;

 free(ptr);

}

void delete_last() {

 node *ptr,*prevptr;

 ptr = start;

 prevptr = start;

 while(ptr->link != NULL) {

prevptr = ptr;

 ptr = ptr->link;

 } prevptr->link = NULL;
```

```
        free(ptr);



}
void delete_val(){
 int val;
 node *ptr,*prevptr;
 prevptr = start;
 ptr = start;
 printf("Enter value to be deleted:\n");
scanf("%d",&val);
 while(ptr->data!=val){
 prevptr = ptr;
 ptr = ptr->link;
 }
 prevptr->link = ptr->link;
 free(ptr);
}
```

```
1.Create 2.Display 3.Delete Head 4.Delete Last 5.Delete val 6.Exit
Enter your choice:
1
Enter element: 10
Do you want to add another element(1/0): 1
Enter element: 20
Do you want to add another element(1/0): 1
Enter element: 30
Do you want to add another element(1/0): 1
Enter element: 40
Do you want to add another element(1/0): 0
1.Create 2.Display 3.Delete Head 4.Delete Last 5.Delete val 6.Exit
Enter your choice:
2
10      20      30      40
1.Create 2.Display 3.Delete Head 4.Delete Last 5.Delete val 6.Exit
Enter your choice:
3
1.Create 2.Display 3.Delete Head 4.Delete Last 5.Delete val 6.Exit
Enter your choice:
2
20      30      40
1.Create 2.Display 3.Delete Head 4.Delete Last 5.Delete val 6.Exit
Enter your choice:
4
1.Create 2.Display 3.Delete Head 4.Delete Last 5.Delete val 6.Exit
Enter your choice:
2
20      30
1.Create 2.Display 3.Delete Head 4.Delete Last 5.Delete val 6.Exit
Enter your choice:
5
Enter value to be deleted:
30
1.Create 2.Display 3.Delete Head 4.Delete Last 5.Delete val 6.Exit
Enter your choice:
2
20
1.Create 2.Display 3.Delete Head 4.Delete Last 5.Delete val 6.Exit
Enter your choice:
```

LAB PROGRAM 7:

WAP to Implement Single Link List with following operations:

a) Sort the linked list.

b) Reverse the linked list.

c) Concatenation of two linked lists.

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

struct NODE
{
   int data;
   struct Node *link;
};
typedef struct NODE node;
node *start = NULL,*start1,*start2,*start3;


node* create()
{
   int choice;
   node *new, *curr;
   start = (node*)malloc(sizeof(node));
   curr = start;
   printf("Enter element:\n");
   scanf("%d", &start->data);
   while(1)
   {
      printf("Do you want to add an element? press 1 for
      yes\n"); scanf("%d", &choice);
      if(choice!=0)
      {
         new = (node*)malloc(sizeof(node));
         printf("Please enter element:\n");
         scanf("%d", &new->data);
         curr->link=new;
         curr = new;
```

```
        }
        else


        {
            curr->link=NULL;
            break;
        }
    }
    return start;
}
void sort()
{
int t,n,count=0,i,j;
node *a,*b,*temp;
temp=start;
while(temp!=NULL)
{
count++;
temp=temp->link;
}
n=count;
a=start;
b=start->link;
for(i=0;i<n-1;i++)
{
for(j=0;j<n-i-1;j++)
{
if(a->data>b->data)
{
t=a->data;
a->data=b->data;
b->data=t;
}
a=b;
b=b->link;
}
a=start;
b=start->link;
}
}
```

```
void reverse()
{
node*a=start,*b=NULL,*c=NULL
; while(a!=NULL)
{
c=b;
b=a;
a=a->link;
b->link=c;
}
start=b;
}

void display()
{
   node *temp;
   temp = start;
   if(start==NULL)
   {
      printf("Linked list is empty\n");
      return;
   }
   while(temp!=NULL)
   {
      printf("%d\t", temp->data);
      temp= temp->link;
   }
}

void concatenate(node *start1,node *start2)
{
node *temp;
if(start1==NULL)
{
start=start2;
return;
}
if(start2==NULL)
{
start=start1;
return;
}
```

```
else
{
temp=start1;


while(temp->link!=NULL)
{
temp=temp->link;
}
temp->link=start2;
start=start1;
}


}

void main()
{
   int choice,c1,c2;
   printf("1.CREATE\n2.SORT\n3.REVERSE\n4.CONCATENATE\n5.DISPLAY\n6.EXIT\n");
   while(1)
   {
      printf("Enter choice:\n");
      scanf("%d", &choice);
      switch(choice)
      {
         case 1: create();
            break;
         case 2: sort();
            break;
         case 3: reverse();
            break;
         case 4: printf("Do ypu want to create the first linked list if yes press 1\n");
               scanf("%d",&c1);
               if(c1==1)
               start1=create();
               else
               start1=NULL;
               printf("Do ypu want to create the second linked list if yes press 2\n");
               scanf("%d",&c2);
               if(c2==2)
               start2=create();
```

```
            else
            start2=NULL;
            concatenate(start1,start2);
        break;


        case 5:display();
        break;
        case 6:exit(0);
        break;
        default: printf("Invalid choice\n");
      }
   }
   getch();
}
```

```
Enter choice:
4
Do ypu want to create the first linked list if yes press 1
1
Enter element:
11
Do you want to add an element? press 1 for yes
1
Please enter element:
22
Do you want to add an element? press 1 for yes
1
Please enter element:
33
Do you want to add an element? press 1 for yes
0
Do ypu want to create the second linked list if yes press 2
2
Enter element:
11
Do you want to add an element? press 1 for yes
1
Please enter element:
77
Do you want to add an element? press 1 for yes
1
Please enter element:
99
Do you want to add an element? press 1 for yes
0
Enter choice:
5
11      22      33      11      77      99      Enter choice:
```

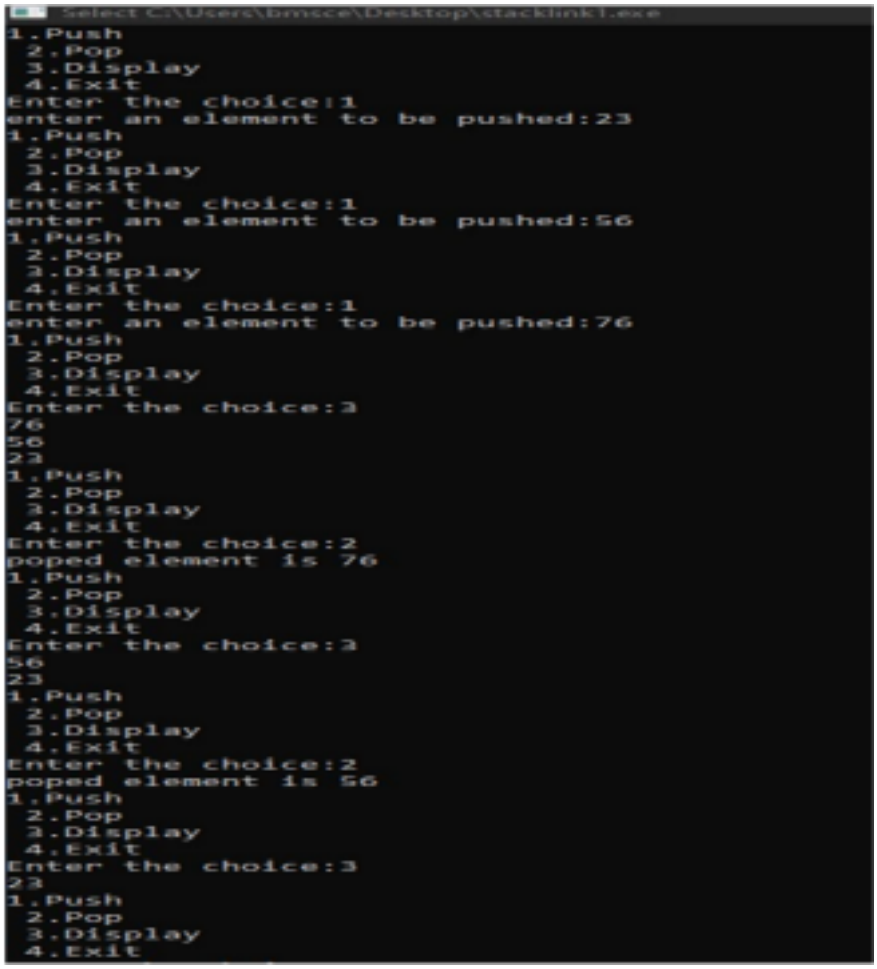
LAB PROGRAM 8:

WAP to implement Stack using Linked Representation.

```
#include<stdio.h>
#include<stdlib.h>
struct NODE
{
   int data;
   struct NODE *link;
};
typedef struct NODE node;
node
*front=NULL,*rear=NULL,*new=NULL; void
disp()
{
   node *temp;
   if(front==NULL)
   {
      printf("Empty");
      return;
   }
   temp=front;
   while(temp!=NULL)
   {
      printf("%d\t",temp->data);
      temp=temp->link;
   }
}
void ins_beg()
{
   new=(node*)malloc(sizeof(node));
   printf("Enter element:");
   scanf("%d",&new->data);
   if(front==NULL)
   {
      front=new;
      rear=new;
      new->link=NULL;
      return;
   }
   new->link=front;
   front=new;
}
```

```
void ins_end()
{
   node *temp;
   temp=rear;
   new=(node*)malloc(sizeof(node))
   ; printf("Enter element:");
   scanf("%d",&new->data);
   if(front==NULL)
   {
      front=new;
      rear=new;
      new->link=NULL;
      return;
   }
   new->link=NULL;
   temp->link=new;
   temp=temp->link;
   rear=temp;
}
void del_beg()
{
   node *temp;
   temp=front;
   if(front==NULL)
   {
      printf("Empty");
      return;
   }
   front=front->link;
   free(temp);
}
void main()
{
   int c1;
   while(1)
   {
      printf("\n1.Push 2.Pop 3.Display
      4.Exit"); printf("\nEnter your choice:");
      scanf("%d",&c1);
      switch(c1)
      {
         case 1:ins_beg();
               break;
         case 2:del_beg();
```

```
            break;
        case 3:disp();


break;
case 4:exit(0);
break;
default:printf("Wrong choice!");
}
}
}
```



WAP to implement Queue using Linked Representation

```
#include<stdio.h>
#include<stdlib.h>
struct NODE
{
   int data;
   struct NODE *link;
};
typedef struct NODE node;
node
*front=NULL,*rear=NULL,*new=NULL; void
disp()
{
   node *temp;
   if(front==NULL)
   {
      printf("Empty");
      return;
   }
   temp=front;
   while(temp!=NULL)
   {
      printf("%d\t",temp->data);
      temp=temp->link;
   }
}
void ins_beg()
{
   new=(node*)malloc(sizeof(node));
   printf("Enter element:");
   scanf("%d",&new->data);
   if(front==NULL)
   {
      front=new;
      rear=new;
      new->link=NULL;
      return;
   }
   new->link=front;
   front=new;
}
void ins_end()
{
```

```
    node *temp;


    temp=rear;
    new=(node*)malloc(sizeof(node));
    printf("Enter element:");
    scanf("%d",&new->data);
    if(front==NULL)
    {
       front=new;
       rear=new;
       new->link=NULL;
       return;
    }
    new->link=NULL;
    temp->link=new;
    temp=temp->link;
    rear=temp;
}
void del_beg()
{
    node *temp;
    temp=front;
    if(front==NULL)
    {
       printf("Empty");
       return;
    }
    front=front->link;
    free(temp);
}
}
void main()
{
    int c1;
    while(1)
    {
       printf("\n1.Insert 2.Delete 3.Display
       4.Exit"); printf("\nEnter your choice:");
       scanf("%d",&c1);
       switch(c1)
       {
          case 1:ins_end();
                 break;
```

```
        case 2:del_beg();
            break;
        case 3:disp();


break;
case 4:exit(0);
break;
default:printf("Wrong choice!");
}
}
}
```

```
C:\Data Structures\Queue-linl

1.Insert 2.Delete 3.Display
Enter your choice:1

Enter the element:5

1.Insert 2.Delete 3.Display
Enter your choice:1

Enter the element:10

1.Insert 2.Delete 3.Display
Enter your choice:3

Queue contains:
5
10

1.Insert 2.Delete 3.Display
Enter your choice:2

Deleted element:5
1.Insert 2.Delete 3.Display
Enter your choice:3

Queue contains:
10

1.Insert 2.Delete 3.Display
Enter your choice:2

Deleted element:10
1.Insert 2.Delete 3.Display
Enter your choice:2

Queue is empty
1.Insert 2.Delete 3.Display
Enter your choice:
```

LAB PROGRAM 9:

WAP to 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.

```
#include<stdio.h>
#include<stdlib.h>
struct NODE
{
struct NODE *llink;
int data;
struct NODE *rlink;
};
typedef struct NODE node;
node *start=NULL,*curr,*new,*temp;
void create()
{
start=(node*)malloc(sizeof(node));
printf("Enter element:");
scanf("%d",&start->data);
start->llink=NULL;
curr=start;
while(1)
{
int choice;
printf("Do you want to add an element? press 1 for yes\n");
      scanf("%d", &choice);
      if(choice!=0)
      {
         new = (node*)malloc(sizeof(node));
         curr->rlink=new;
         new->llink=curr;
         printf("Enter the element:");
         scanf("%d",&new->data);
         curr=new;
   }
   Else
   {
```

```
    curr->rlink=NULL;
    break;
}
}
}
void insert_beg()
{
 new=(node*)malloc(sizeof(node))
 ; printf("Enter an element:");
scanf("%d",&new->data);
if(start==NULL)
{
 new->llink=NULL;
 new->rlink=NULL;
start=new;
 return;
}
 new->rlink=start;
start->llink=new;
 new->llink=NULL;
start=new;
}
void delete_ele()
{node *temp;
int ele;
if(start==NULL)
{
 printf("Linked list is
 empty\n"); return;
}
 printf("Enter element to be
deleted:"); scanf("%d",&ele);
if(start->data==ele)
{
temp=start;
start=start->rlink;
start->llink=NULL;
free(temp);
 return;
}
```

```
temp=start;
while(temp->rlink!=NULL&&temp->data!=ele)
{
temp=temp->rlink;
}
if(temp->data==ele&&temp->rlink==NULL)
{
temp->llink->rlink=NULL;
free(temp);
return;
}
if(temp->data==ele&&temp->rlink!=NULL)
{
temp->llink->rlink=temp->rlink;
temp->rlink->llink=temp->llink;
free(temp);
return;
}
printf("Element not found\n");
}

void display()
{
if(start==NULL)
{
printf("Linked list is empty\n");
return;
}
temp=start;
while(temp!=NULL)
{
printf("%d\t",temp->data);
temp=temp->rlink;
}
}

void main()
{
   int choice;
   printf("1.CREATE\n2.INSERT AT BEGINING\n3.DELETE SPECIFIC
ELEMENT\n4.DISPLAY\n5.EXIT\n");
```

```
  while(1)
  {
     printf("Enter choice:\n");
     scanf("%d", &choice);
     switch(choice)
     {
        case 1: create();
           break;
        case 2: insert_beg();
           break;
        case 3: delete_ele();
           break;
        case 4:display();
        break;
        case 5:exit(0);
        break;
        default:printf("Invalid choice\n");
     }
  }
  getch();
}
```

```
1.Create
 2.Display
 3.Insert
 4.Delete the specific value
 5.Exit
Enter the choice:1
Enter an element45
Do you want to enter a new element (1 for yes,any other number for no)1
Enter an element67
Do you want to enter a new element (1 for yes,any other number for no)1
Enter an element87
Do you want to enter a new element (1 for yes,any other number for no)234
1.Create
 2.Display
 3.Insert
 4.Delete the specific value
 5.Exit
Enter the choice:2
45
67
87
1.Create
 2.Display
 3.Insert
 4.Delete the specific value
 5.Exit
Enter the choice:3
Enter an element89
1.Create
 2.Display
 3.Insert
 4.Delete the specific value
 5.Exit
Enter the choice:2
89
45
67
87
1.Create
 2.Display
 3.Insert
 4.Delete the specific value
 5.Exit
Enter the choice:4
Enter the element to be deleted:45
1.Create
 2.Display
 3.Insert
 4.Delete the specific value
 5.Exit
Enter the choice:2
89
67
87
1.Create
 2.Display
 3.Insert
 4.Delete the specific value
 5.Exit
```