DS LAB RECORD

10 EXPERIMENTS

Mallika Prasad

1BM19CS081

LAB 1-

WAP 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 and stack underflow.

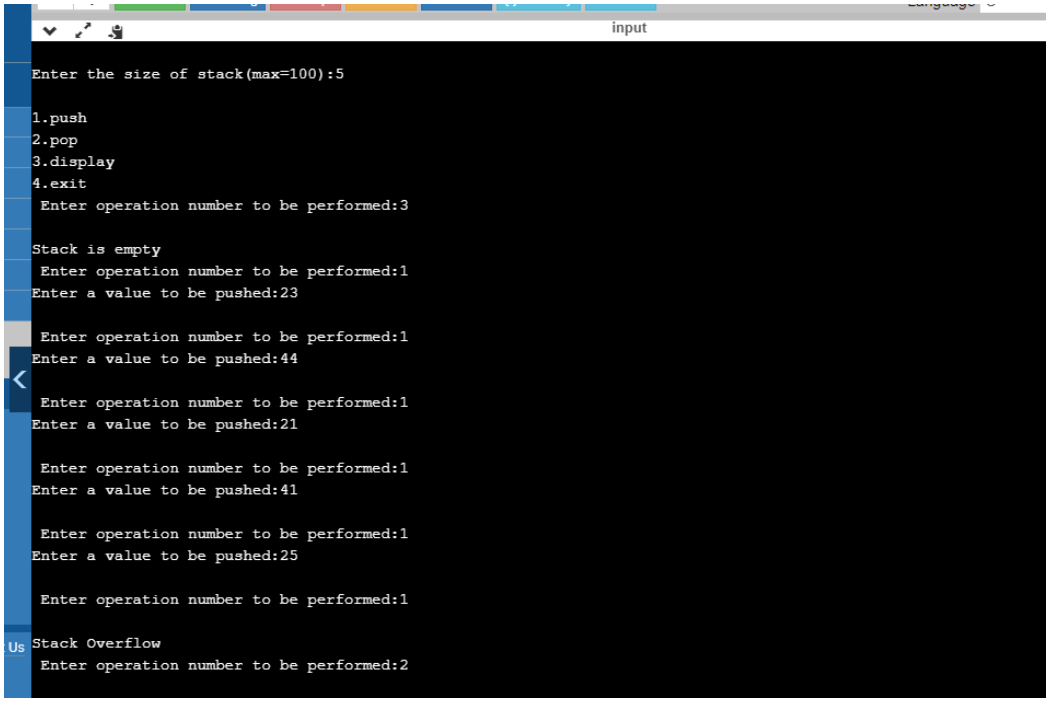
```
#include<stdio.h>

void push();

void pop();

void display();

int stack[100],choice,n,top,x,i;

int main()

{

 top=-1;

 printf("\nEnter the size of stack(max=100):");

 scanf("%d",&n);

 printf("\n1.push\n2.pop\n3.display\n4.exit");

 do

 {

 printf("\n Enter operation number to be performed:");

 scanf("%d",&choice);

 switch(choice)
```

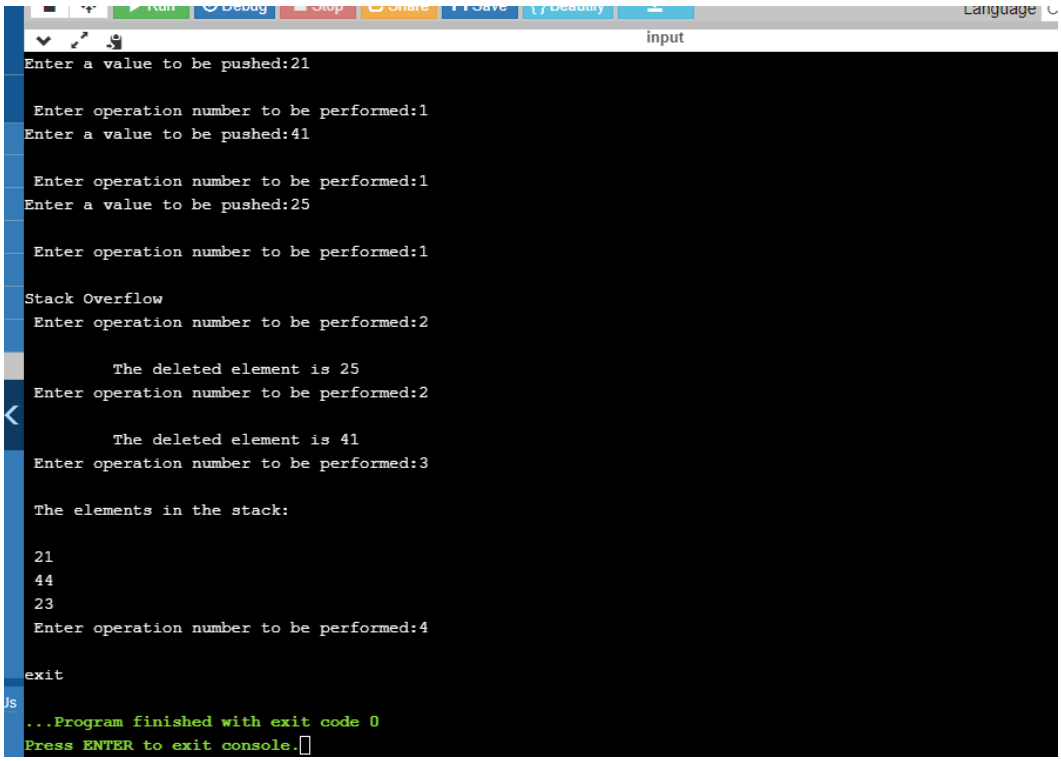
```
{
case 1:
{
push();
break;
}
case 2:
{
pop();
break;
}
case 3:
{
display();
break;
}
case 4:
{
printf("\nexit");
break;
}
default:
{
printf ("\ninvalid Choice");
```

```
 }
 }
 }
 while(choice!=4);
 return 0;
}
void push()
{
 if(top>=n-1)
 {
 printf("\nStack Overflow");

 }
 else
 {
 printf("Enter a value to be pushed:");
 scanf("%d",&x);
 top++;
 stack[top]=x;
 }
}
void pop()
{
 if(top<=-1)
```

```
 {
 printf("\nStack underflow/empty");
 }
 else
 {
 printf("\n\t The deleted element is %d",stack[top]);
 top--;
 }
}
void display()
{
 if(top>=0)
 {
 printf("\n The elements in the stack: \n");
 for(i=top; i>=0; i--)
 printf("\n> %d",stack[i]);
 }
 else
 {
 printf("\nStack is empty");
 }
}
```

Output-


input
Enter the size of stack(max=100):5
1.push
2.pop
3.display
4.exit
Enter operation number to be performed:3
Stack is empty
Enter operation number to be performed:1
Enter a value to be pushed:23
Enter operation number to be performed:1
Enter a value to be pushed:44
Enter operation number to be performed:1
Enter a value to be pushed:21
Enter operation number to be performed:1
Enter a value to be pushed:41
Enter operation number to be performed:1
Enter a value to be pushed:25
Enter operation number to be performed:1
Stack Overflow
Enter operation number to be performed:2


input
Enter a value to be pushed:21
Enter operation number to be performed:1
Enter a value to be pushed:41
Enter operation number to be performed:1
Enter a value to be pushed:25
Enter operation number to be performed:1
Stack Overflow
Enter operation number to be performed:2
The deleted element is 25
Enter operation number to be performed:2
The deleted element is 41
Enter operation number to be performed:3
The elements in the stack:
21
44
23
Enter operation number to be performed:4
exit
...Program finished with exit code 0
Press ENTER to exit console.

LAB 2-

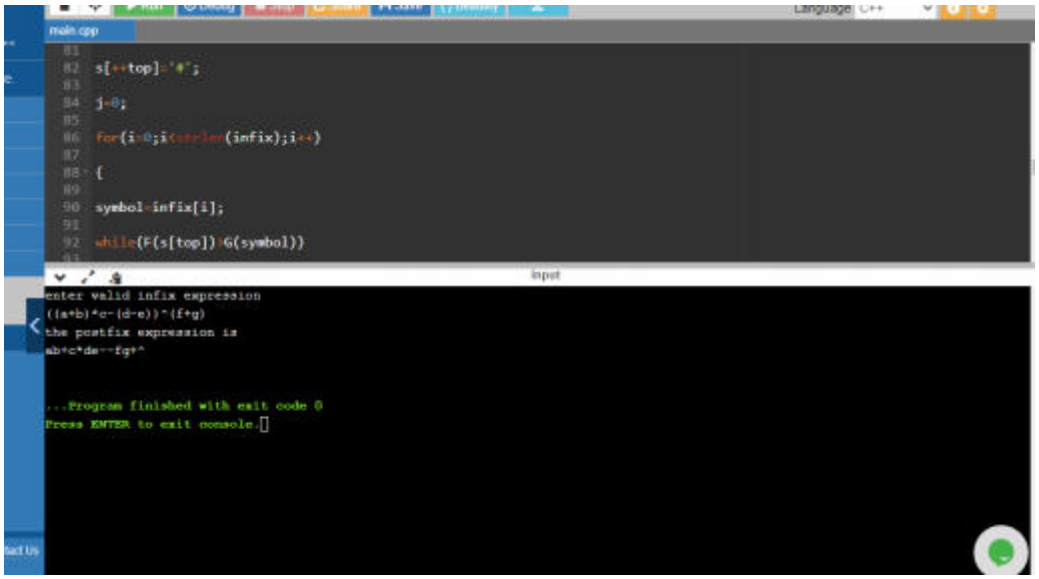
WAP to convert a given valid parenthesized infix arithmetic expression to postflix expression. The expression consists of single character operands and the binary operators plus +, minus -, multiply * and divide /.

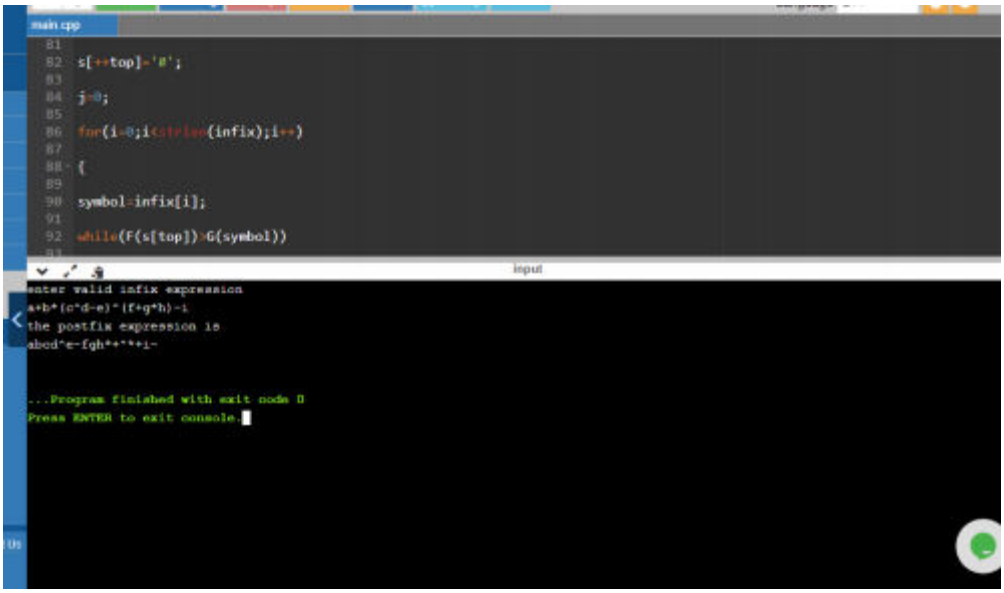
```
#include<stdio.h>

#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], symbol;
top=-1;
s[++top]='#';
j=0;
for(i=0;i<strlen(infix);i++)
{
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';
}
int main ()
{
char infix[20];
char postfix[20];
printf("enter valid infix expression \n");
scanf("%s",infix);
infix_postfix(infix,postfix);
printf("the postfix expression is \n");
printf("%s\n", postfix);
return 0;
}
```

Output-


main.cpp
s[++top]='#';
j=0;
for(i=0;i<strlen(infix);i++)
{
symbol=infix[i];
while(F(s[top])>G(symbol))
Input
enter valid infix expression
((a+b)*c-(d-e))^(f+g)
the postfix expression is
ab+c*de--fg+^
...Program finished with exit code 0
Press ENTER to exit console.


main.cpp
s[++top]='#';
j=0;
for(i=0;i<strlen(infix);i++)
{
symbol=infix[i];
while(F(s[top])>G(symbol))
Input
enter valid infix expression
a+b*(c*d-e)^(f+g*h)-i
the postfix expression is
abcd*e-fgh*+^*+i-
...Program finished with exit code 0
Press ENTER to exit console.

LAB 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 of queue empty and queue overflow condition.

```
#include <stdio.h>

#define que_size 3

int item,front=0,rear=-1,q[10];

void insertrear()

{

 if(rear==que_size-1)

 {

 printf("QUEUE OVERFLOW\n");

 return;

 }

 rear=rear+1;

 q[rear]=item;

}

int deletefront()

{

 if(front>rear)

 {

 front=0;

 rear=-1;

 return -1;

 }
```

```
 return q[front++];
}
void displayQ()
{
 int i;
 if(front>rear)
 {
 printf("QUEUE IS EMPTY\n");
 return;
 }
 printf("contents of the queue\n");
 for(i=front;i<=rear;i++)
 {
 printf("%d\n",q[i]);
 }
}
int main()
{
 int choice;
 do
 {
 printf("\n1.insert rear\n2.delete front\n3.display\n4.exit\n");
 printf("enter the choice\n");
 scanf("%d",&choice);
```

```
switch(choice)
{
case 1:printf("enter the item to be inserted\n");
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:break;
}
}
while(choice!=4);
return 0;
}
```

Output-

```
1.insert rear
2.delete front
3.display
4.exit
enter the choice
1
enter the item to be inserted
12

1.insert rear
2.delete front
3.display
4.exit
enter the choice
1
enter the item to be inserted
13

1.insert rear
2.delete front
3.display
4.exit
enter the choice
1
enter the item to be inserted
14

1.insert rear
2.delete front
3.display
```

```
1.insert rear
2.delete front
3.display
4.exit
enter the choice
1
enter the item to be inserted
15
QUEUE OVERFLOW

1.insert rear
2.delete front
3.display
4.exit
enter the choice
3
contents of the queue
12
13
14

1.insert rear
2.delete front
3.display
4.exit
enter the choice
2
item deleted is 12

1.insert rear
2.delete front
```

```
3.display
4.exit
enter the choice

2
item deleted is 13

1.insert rear
2.delete front
3.display
4.exit
enter the choice
2
item deleted is 14

1.insert rear
2.delete front
3.display
4.exit
enter the choice
3
QUEUE IS EMPTY

1.insert rear
2.delete front
3.display
4.exit
enter the choice
4
```

```
enter the choice

2
item deleted is 13

1.insert rear
2.delete front
3.display
4.exit
enter the choice
2
item deleted is 14

1.insert rear
2.delete front
3.display
4.exit
enter the choice
3
QUEUE IS EMPTY

1.insert rear
2.delete front
3.display
4.exit
enter the choice
4


...Program finished with exit code 0
Press ENTER to exit console.
```

LAB 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>

#define QUE_SIZE 3

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

void insertrear()

{

if(count==QUE_SIZE)

{

printf("queue overflow\n");

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;
```

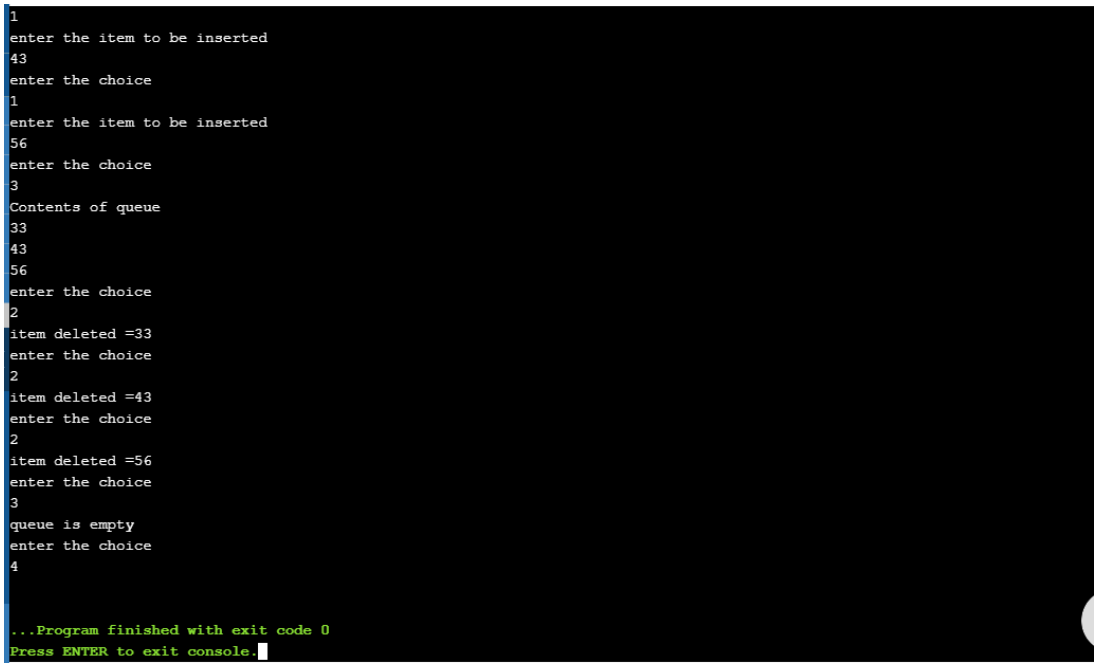
```
}
void displayQ()
{
int i,f;
if(count==0)
{
printf("queue is empty\n");
return;
}
f=front;
printf("Contents of queue \n");
for(i=1;i<=count;i++)
{
printf("%d\n",q[f]);
f=(f+1)%QUE_SIZE;
}
}
int main()
{
 int choice;


 printf("\n1:insertrear\n2:deletefront\n3:display\n4:exit\n");
do
```

```
{
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item to be inserted\n");
scanf("%d",&item);
insertrear();
break;
case 2:item=deletefront();
if(item==-1)
printf("queue is empty\n");
else
printf("item deleted =%d\n",item);
break;
case 3:displayQ();
break;
default:break;
}
}
while(choice!=4);
return 0;
}
```

Output-

```
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
11
enter the choice
1
enter the item to be inserted
12
enter the choice
1
enter the item to be inserted
13
enter the choice
1
enter the item to be inserted
14
queue overflow
enter the choice
3
Contents of queue
11
12
13
enter the choice
2
item deleted =11
```

```
2
item deleted =11
enter the choice
3
Contents of queue
12
13
enter the choice
1
enter the item to be inserted
33
enter the choice
3
Contents of queue
12
13
33
enter the choice
2
item deleted =12
enter the choice
2
item deleted =13
enter the choice
1
enter the item to be inserted
43
enter the choice
1
enter the item to be inserted
56
```



LAB 5-

WAP to implement Singly Linked List with the following operations a]create a linked list b]Insertion of node at first position and end of the list c] deletion of node at first position and end of the list d]displaying contents of the list

```
#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("mem 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("iten deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
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);
 }
}
int main()
{
int item,choice;
NODE first=NULL;
printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n
5:display_list\n6:Exit\n");
do
```

```
{ printf("\nenter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
 case 1:printf("enter the item at front-end\n");
 scanf("%d",&item);
 first=insert_front(first,item);
 break;
 case 2:first=delete_front(first);
 break;
 case 3:printf("enter the item at rear-end\n");
 scanf("%d",&item);
 first=insert_rear(first,item);
 break;
 case 4:first=delete_rear(first);
 break;
 case 5:display(first);
 break;
 case 6:break;
 default:break;
 }
}while(choice!=6);
return 0;
}
```

Output-

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:display_list
6:Exit

enter the choice
1
enter the item at front-end
11

enter the choice
1
enter the item at front-end
12

enter the choice
1
enter the item at front-end
13

enter the choice
5
13
12
11

enter the choice
```

```
enter the choice
3
enter the item at rear-end
14

enter the choice
3
enter the item at rear-end
15

enter the choice
3
enter the item at rear-end
16

enter the choice
5
13
12
11
14
15
16

enter the choice
2
item deleted at front-end is=13

enter the choice
4
iten deleted at rear-end is 16
```

```
iten deleted at rear-end is 16
enter the choice
4
iten deleted at rear-end is 15
enter the choice
4
iten deleted at rear-end is 14
enter the choice
5
12
11

enter the choice
2
item deleted at front-end is=12

enter the choice
2
item deleted at front-end is=11

enter the choice
5
list empty cannot display items

enter the choice
2
list is empty cannot delete

enter the choice
4
```

```
enter the choice
2
item deleted at front-end is=12

enter the choice
2
item deleted at front-end is=11

enter the choice
5
list empty cannot display items

enter the choice
2
list is empty cannot delete

enter the choice
4
list is empty cannot delete

enter the choice
6


...Program finished with exit code 0
Press ENTER to exit console.
```

LAB 6-

WAP to implement Single Linked List with the following operations a]create a link list b]deletion of first element, specified element and last element c]insertion of a node at first position, any position and last element d]display the contents of the list.

```
#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("mem 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("iten 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 NULL;
}
if (pos==1)
{
temp->link=first;
return temp;
}
count=1;
prev=NULL;
cur=first;
while (cur!=NULL && count!=pos)
{
prev=cur;
cur=cur->link;
```

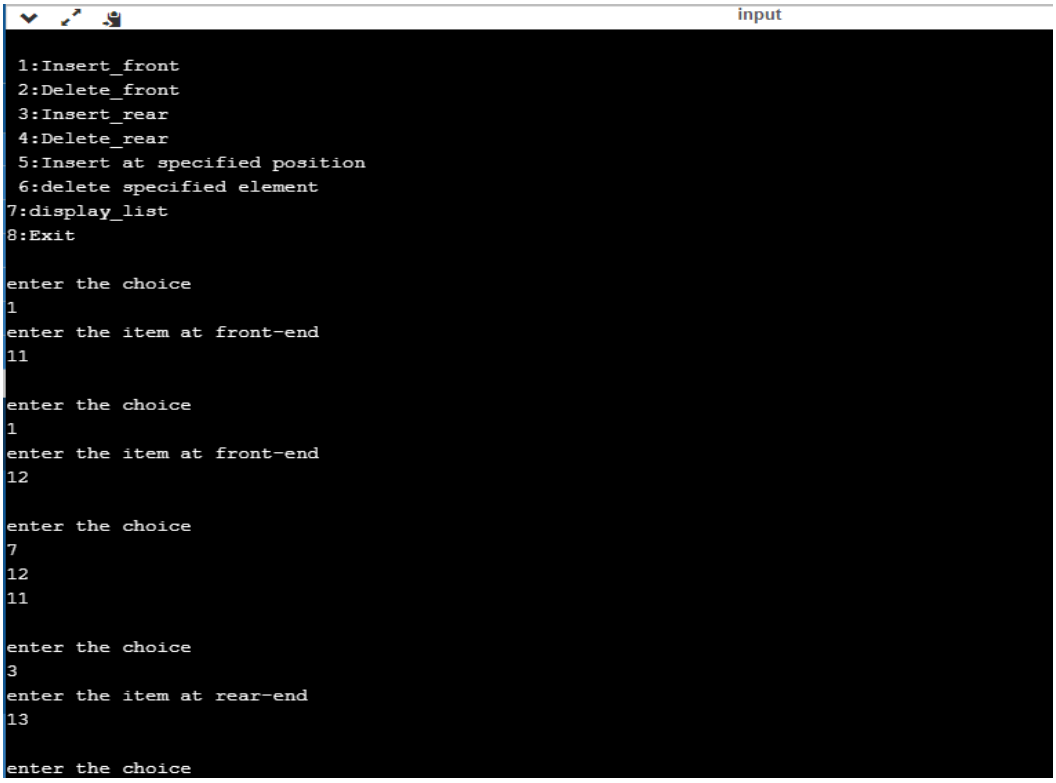
```
 count++;
 }
 if (count==pos)
 {
 prev->link=temp;
 temp->link=cur;
 return first;
 }
 printf("Invalid position\n");
 return first;
}
NODE delete_info(int item,NODE first)
{
NODE prev,cur;
if(first==NULL)
{
printf("list is empty\n");
return NULL;
}
if(item==first->info)
{
cur=first;
first=first->link;
freenode(cur);
```

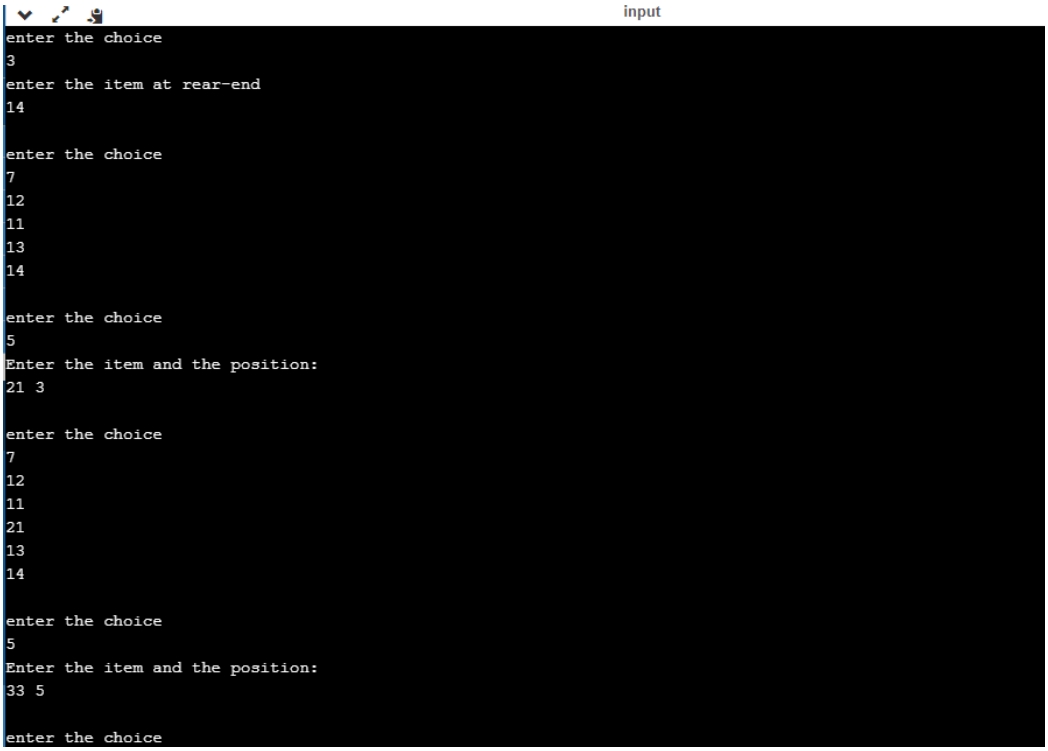
```
return first;
}
prev=NULL;
cur=first;
while(cur!=NULL)
{
if(item==cur->info)break;
prev=cur;
cur=cur->link;
}
if(cur==NULL)
{
printf("search is unsuccessfull\n");
return first;
}
prev->link=cur->link;
printf("item deleted is %d",cur->info);
freenode(cur);
return first;
}
void display(NODE first)
{
 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);
 }
}
int main()
{
int item,choice,pos;
NODE first=NULL;
printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:Insert at
specified position\n 6:delete specified element \n7:display_list\n8:Exit\n");
do
{
printf("\nenter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
 case 1:printf("enter the item at front-end\n");
 scanf("%d",&item);
 first=insert_front(first,item);
 break;
 case 2:first=delete_front(first);
 break;
```

```
 case 3:printf("enter the item at rear-end\n");
 scanf("%d",&item);
 first=insert_rear(first,item);
 break;
 case 4:first=delete_rear(first);break;
 case 5:printf("Enter the item and the position:\n");
 scanf("%d%d",&item,&pos);
 first=insert_pos(item,pos,first);
 break;
 case 6:printf("enter the element to be deleted\n");
 scanf("%d",&item);
 first=delete_info(item,first);
 break;
 case 7:display(first);
 break;
 case 8:break;
 default:break;
 }
}while(choice!=8);
return 0;
}
```

Output-


input
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Insert at specified position
6:delete specified element
7:display_list
8:Exit
enter the choice
1
enter the item at front-end
11
enter the choice
1
enter the item at front-end
12
enter the choice
7
12
11
enter the choice
3
enter the item at rear-end
13
enter the choice

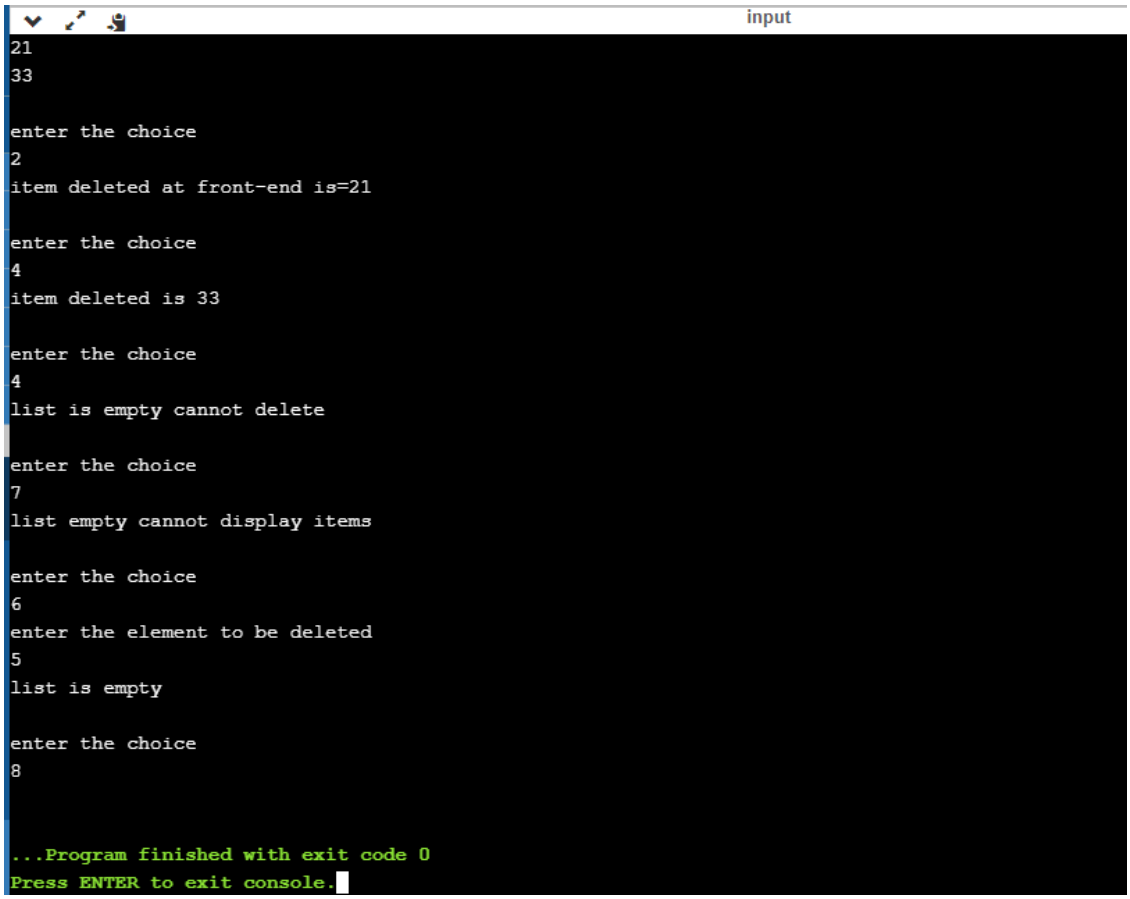

input
enter the choice
3
enter the item at rear-end
14
enter the choice
7
12
11
13
14
enter the choice
5
Enter the item and the position:
21 3
enter the choice
7
12
11
21
13
14
enter the choice
5
Enter the item and the position:
33 5
enter the choice

input

```
enter the choice
7
12
11
21
13
33
14

enter the choice
6
enter the element to be deleted
13
item deleted is 13
enter the choice
6
enter the element to be deleted
11
item deleted is 11
enter the choice
7
12
21
33
14

enter the choice
2
item deleted at front-end is=12

enter the choice
```

input

```
enter the choice
2
item deleted at front-end is=12

enter the choice
4
iten deleted at rear-end is 14
enter the choice
7
21
33

enter the choice
2
item deleted at front-end is=21

enter the choice
4
item deleted is 33

enter the choice
4
list is empty cannot delete

enter the choice
7
list empty cannot display items

enter the choice
6
enter the element to be deleted
```



LAB 7-

WAP to implement Single Linked List with the following operations a] sort the linked list b]reverse the linked list c]concatenation of 2 linked lists

```
#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("mem 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("iten deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}
NODE order_list(int item,NODE first)
{
NODE temp,prev,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL) return temp;
if(item<first->info)
{
temp->link=first;
return temp;
}
prev=NULL;
cur=first;
```

```
while(cur!=NULL&&item>cur->info)
{
prev=cur;
cur=cur->link;
}
prev->link=temp;
temp->link=cur;
return first;
}
NODE reverse(NODE first)
{
NODE cur,temp;
cur=NULL;
while(first!=NULL)
{
temp=first;
first=first->link;
temp->link=cur;
cur=temp;
}
return cur;
}
NODE concat(NODE first,NODE second)
{
```

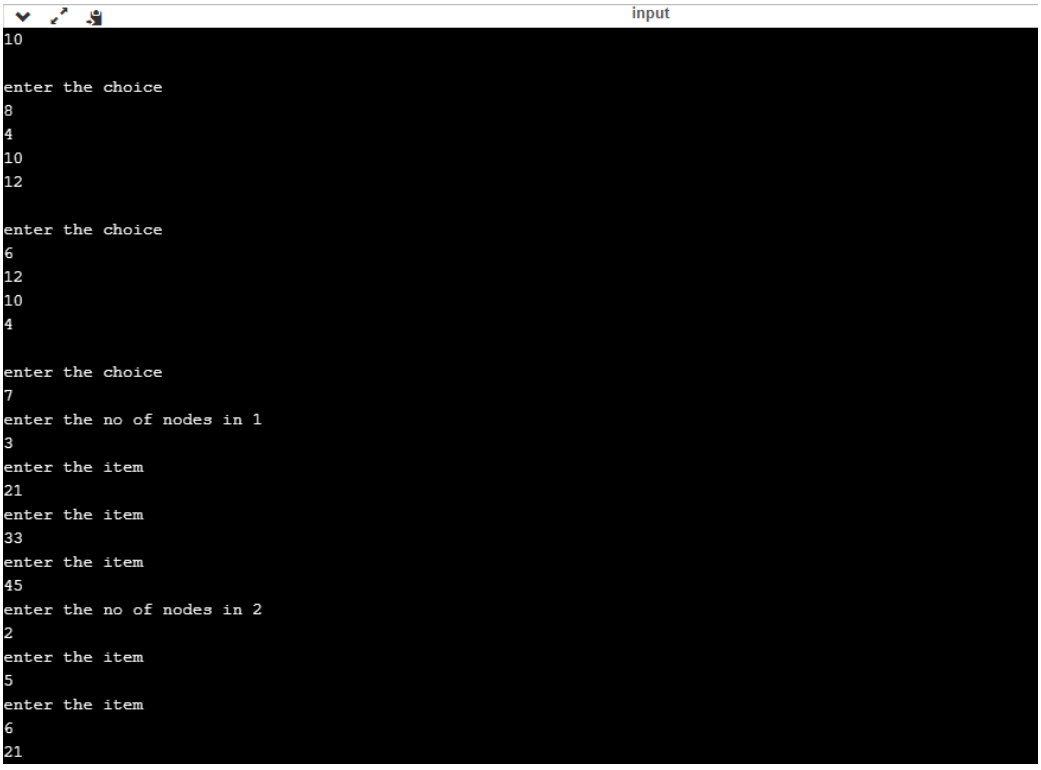
```
NODE cur;
if(first==NULL)
return second;
if(second==NULL)
return first;
cur=first;
while(cur->link!=NULL)
cur=cur->link;
cur->link=second;
return first;
}
void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("list empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
 {
 printf("%d\n",temp->info);
 }
}
int main()
{
int item,choice,pos,n;
```

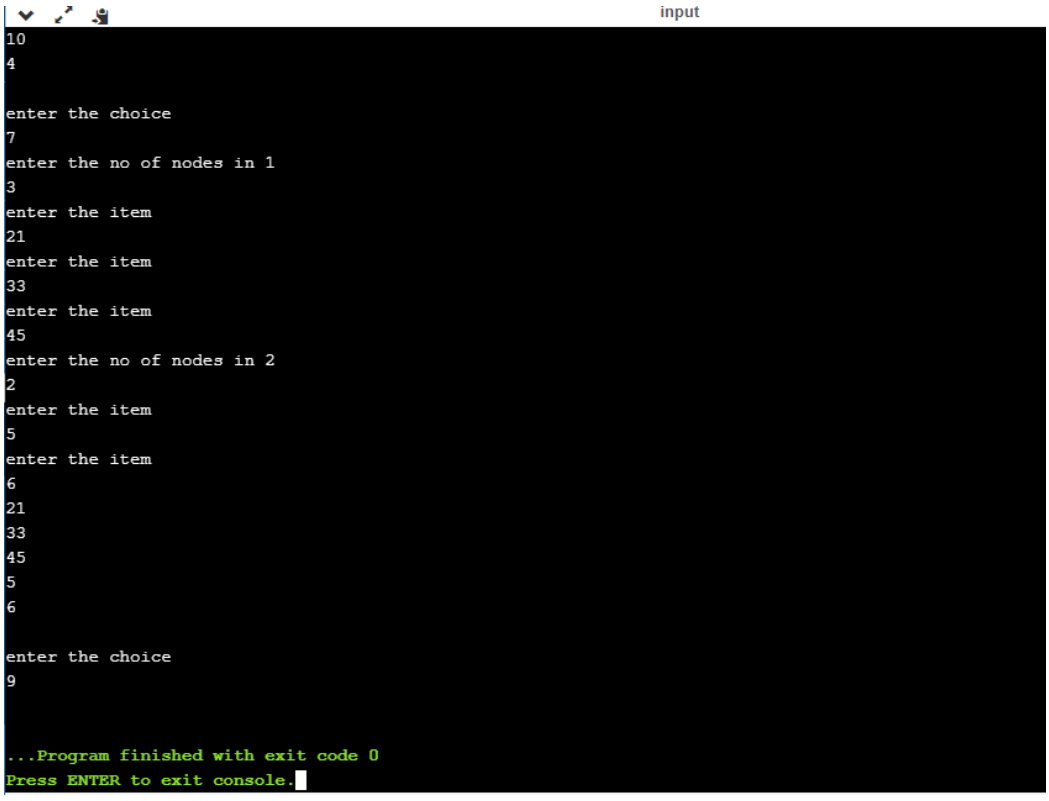
```
NODE first=NULL,a,b;
printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:sorted list \n
6:reverse the list \n7:concatinate 2 strings\n 8:display_list\n9:Exit\n");
do
{
printf("\nenter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
 case 1:printf("enter the item at front-end\n");
 scanf("%d",&item);
 first=insert_front(first,item);
 break;
 case 2:first=delete_front(first);
 break;
 case 3:printf("enter the item at rear-end\n");
 scanf("%d",&item);
 first=insert_rear(first,item);
 break;
 case 4:first=delete_rear(first);break;
 case 5:printf("enter the item to be inserted in ordered_list\n");
 scanf("%d",&item);
 first=order_list(item,first);
 break;
```

```
case 6:first=reverse(first);
display(first);
break;
case 7:printf("enter the no of nodes in 1\n");
scanf("%d",&n);
a=NULL;
for(int i=0;i<n;i++)
{
printf("enter the item\n");
scanf("%d",&item);
a=insert_rear(a,item);
}
printf("enter the no of nodes in 2\n");
scanf("%d",&n);
b=NULL;
for(int i=0;i<n;i++)
{
printf("enter the item\n");
scanf("%d",&item);
b=insert_rear(b,item);
}
a=concat(a,b);
display(a);
break;
```

```
 case 8:display(first);
 break;
 case 9:break;
 default:break;
 }
}while(choice!=9);
return 0;
}
```

Output-


input
10
enter the choice
8
4
10
12
enter the choice
6
12
10
4
enter the choice
7
enter the no of nodes in 1
3
enter the item
21
enter the item
33
enter the item
45
enter the no of nodes in 2
2
enter the item
5
enter the item
6
21


input
10
4
enter the choice
7
enter the no of nodes in 1
3
enter the item
21
enter the item
33
enter the item
45
enter the no of nodes in 2
2
enter the item
5
enter the item
6
21
33
45
5
6
enter the choice
9
...Program finished with exit code 0
Press ENTER to exit console.

LAB 8-

WAP to implement stack and queues using linked representation

```
#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("mem full\n");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
```

```
}
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("stack 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\n",cur->info);
free(cur);
prev->link=NULL;
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;
}
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);
}
}
int main()
{
int item,choice;
NODE first=NULL;
 printf("STACK-insert rear and delete rear\nQUEUE-insert rear and delete front\n");
printf("\n 1:Insert_rear\n 2:Delete_rear\n 3:Delete_front\n 4:Display_list\n 5:Exit\n");
do
```

```
{
printf("enter the choice\n");
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:first=delete_front(first);
break;
case 4:display(first);
break;
case 5:break;
default:printf("invalid choice");
break;
}
}while(choice!=5);
return 0;
}
```

Output-

```
STACK-insert rear and delete rear
QUEUE-insert rear and delete front

 1:Insert_rear
 2:Delete_rear
 3:Delete_front
 4:Display_list
 5:Exit
enter the choice
1
enter the item at rear-end
11
enter the choice

1
enter the item at rear-end
12
enter the choice
1
enter the item at rear-end
13
enter the choice
1
enter the item at rear-end
14
enter the choice
4
11
12
13
14
```

```
13
14
enter the choice
2
item deleted at rear-end is 14
enter the choice
2
item deleted at rear-end is 13
enter the choice
4
11
12
enter the choice
2
item deleted at rear-end is 12
enter the choice
2
item deleted is 11
enter the choice
2
stack is empty cannot delete
enter the choice
1
enter the item at rear-end
22
enter the choice
1
enter the item at rear-end
33
enter the choice
1
```

```
33
enter the choice
1
enter the item at rear-end
44
enter the choice
1
enter the item at rear-end
55
enter the choice
4
22
33
44
55
enter the choice
3
item deleted at front-end is=22
enter the choice
3
item deleted at front-end is=33
enter the choice
4
44
55
enter the choice
3
item deleted at front-end is=44
enter the choice
3
item deleted at front-end is=55
```

```
4
22
33
44
55
enter the choice
3
item deleted at front-end is=22
enter the choice
3
item deleted at front-end is=33
enter the choice
4
44
55
enter the choice
3
item deleted at front-end is=44
enter the choice
3
item deleted at front-end is=55
enter the choice
3
list is empty cannot delete
enter the choice
5


...Program finished with exit code 0
Press ENTER to exit console.
```

LAB 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 e) Delete the duplicates [plus other functions]

```
#include<stdio.h>
#include<stdlib.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("mem full\n");
 exit(0);
 }
 return x;
 }
void freenode(NODE x)
```

```
{
 free(x);
}
NODE dinsert_front(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
cur=head->rlink;
head->rlink=temp;
temp->llink=head;
temp->rlink=cur;
cur->llink=temp;
return head;
}
NODE dinsert_rear(int item,NODE head)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
cur=head->llink;
head->llink=temp;
temp->rlink=head;
temp->llink=cur;
```

```
cur->rlink=temp;
return head;
}
NODE ddelete_front(NODE head)
{
NODE cur,next;
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
cur=head->rlink;
next=cur->rlink;
head->rlink=next;
next->llink=head;
printf("the node deleted is %d\n",cur->info);
freenode(cur);
return head;
}
NODE ddelete_rear(NODE head)
{
NODE cur,prev;
if(head->rlink==head)
{
```

```
printf("list empty\n");
return head;
}
cur=head->llink;
prev=cur->llink;
head->llink=prev;
prev->rlink=head;
printf("the node deleted is %d\n",cur->info);
freenode(cur);
return head;
}
void display(NODE head)
{
NODE temp;
if(head->rlink==head)
{
printf("list empty\n");
return;
}
printf("contents of list\n");
temp=head->rlink;
while(temp!=head)
{
printf("%d\n",temp->info);
```

```
temp=temp->rlink;
}
printf("\n");
}
NODE insert_leftpos(int item,NODE head)
{
NODE temp,cur,prev;
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)break;
cur=cur->rlink;
}
if(cur==head)
{
 printf("key not found\n");
 return head;
 }
 prev=cur->llink;
```

```
 printf("enter towards left of %d=",item);
 temp=getnode();
 scanf("%d",&temp->info);
 prev->rlink=temp;
 temp->llink=prev;
 cur->llink=temp;
 temp->rlink=cur;
 return head;
}
NODE insert_rightpos(int item,NODE head)
{
NODE temp,cur,next;
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)break;
cur=cur->rlink;
}
if(cur==head)
```

```
{
 printf("key not found\n");
 return head;
 }
 next=cur->rlink;
 printf("enter towards right of %d=",item);
 temp=getnode();
 scanf("%d",&temp->info);
 next->llink=temp;
 temp->rlink=next;
 cur->rlink=temp;
 temp->llink=cur;
 return head;
}
NODE delete_dup_key(int item,NODE head)
{
NODE prev, cur, next;
 int count;
 if (head->rlink == head)
 {
 printf("List is Empty!");
 return head;
 }
 count = 0;
```

```
cur = head->rlink;
while (cur != head)
{
if (item != cur->info)
cur = cur->rlink;
else
{
count++;
if(count==1)
{
cur = cur->rlink;
}
if(count!=1)
{
prev = cur->llink;
next = cur->rlink;
prev->rlink = next;
next->llink = prev;
freenode(cur);
cur = next;
}


}
}
```

```
 if (count == 0)
 printf("Key not found");
 else
 printf("Duplicates are deleted\n");
 return head;
}
void search(int item,NODE head){
 NODE cur;
 if(head->rlink==head)
 {
 printf("List Empty");
 return ;
 }
 cur=head->rlink;
 while(cur!=head)
 {
 if(item==cur->info)break;
 cur=cur->rlink;
 }
 if(cur==head)
 {printf("search unsuccessfull\n");
 return;
 }
 printf("search successful\n");
```

```
}
NODE delete_value(int item,NODE head)
{
NODE prev,cur,next;
int count;
 if(head->rlink==head)
 {
 printf("List Empty");
 return head;
 }
count=0;
cur=head->rlink;
while(cur!=head)
{
 if(item!=cur->info)
 cur=cur->rlink;
 else
 {
 count++;
 prev=cur->llink;
 next=cur->rlink;
 prev->rlink=next;
 next->llink=prev;
```

```
 freenode(cur);
 }
}
if(count==0)
 printf("key not found\n");
 else
 printf("key found at %d positions and are deleted\n",count);
return head;
}
int main()
{
NODE head,last;
int item, choice;
head=getnode();
head->rlink=head;
head->llink=head;
 printf("\n1:insert front\n2:insert rear\n3:delete front\n4:delete rear\n5:insert key
towards left\n6:insert key towards right\n7:Delete duplicate keys\n8:delete all specified
node\n9:search item\n10:display\n11:exit\n");
do
 {
 printf("enter the choice\n");
 scanf("%d",&choice);
 switch(choice)
```

```
{
case 1: printf("enter the item at front end\n");
scanf("%d",&item);
last=dinsert_front(item,head);
break;
case 2: printf("enter the item at rear end\n");
scanf("%d",&item);
last=dinsert_rear(item,head);
break;
case 3:last=ddelete_front(head);
break;
case 4: last=ddelete_rear(head);
break;
case 5: printf("enter the key item\n");
scanf("%d",&item);
last=insert_leftpos(item,head);
break;
case 6: printf("enter the key item\n");
scanf("%d",&item);
last=insert_rightpos(item,head);
break;
case 7:printf("enter key whose duplicates to be deleted\n");
scanf("%d",&item);
last=delete_dup_key(item,head);
```

```
 break;
 case 8:printf("enter key which has to be deleted\n");
 scanf("%d",&item);
 last=delete_value(item,head);
 break;
 case 9:printf("enter item\n");
 scanf("%d",&item);
 search(item,head);
 break;
 case 10: display(head);
 break;
 case 11:break;
 default:printf("invalid choice");
 break;
 }
 }while(choice!=11);
return 0;
}
```

Output-

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:insert key towards left
6:insert key towards right
7:Delete duplicate keys
8:delete all specified node
9:search item
10:display
11:exit
enter the choice
1
enter the item at front end
11
enter the choice
1
enter the item at front end
12
enter the choice
1
enter the item at front end
13
enter the choice
2
enter the item at rear end
21
enter the choice
2
enter the item at rear end
```

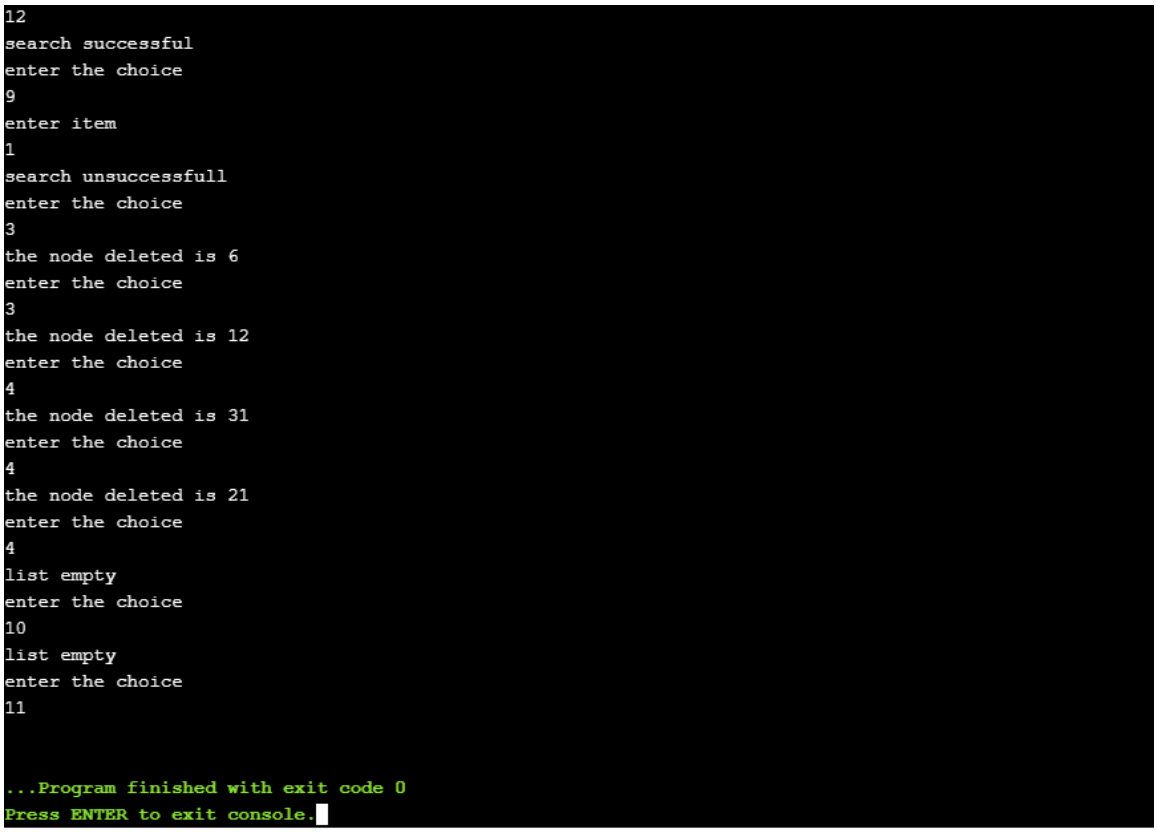
```
enter the item at rear end
31
enter the choice
2
enter the item at rear end
41
enter the choice
10
contents of list
13
12
11
21
31
41

enter the choice
3
the node deleted is 13
enter the choice
4
the node deleted is 41
enter the choice
10
contents of list
12
11
21
31

enter the choice
```

```
enter the choice
5
enter the key item
11
enter towards left of 11=6
enter the choice
6
enter the key item
21
enter towards right of 21=7
enter the choice
10
contents of list
12
6
11
21
7
31

enter the choice
1
enter the item at front end
6
enter the choice
1
enter the item at front end
6
enter the choice
2
enter the item at rear end
```

```
enter the item at rear end
6
enter the choice
10
contents of list
6
6
12
6
11
21
7
31
6

enter the choice
7
enter key whose duplicates to be deleted
6
Duplicates are deleted
enter the choice
10
contents of list
6
12
11
21
7
31

enter the choice
```

```
enter the choice
1
enter the item at front end
11
enter the choice
1
enter the item at front end
11
enter the choice
2
enter the item at rear end
11
enter the choice
10
contents of list
11
11
6
12
11
21
7
31
11

enter the choice
8
enter key which has to be deleted
7
key found at 1 positions and are deleted
enter the choice
```

```
enter the choice
8
enter key which has to be deleted
11
key found at 4 positions and are deleted
enter the choice
10
contents of list
6
12
21
31

enter the choice
9
enter item
12
search successful
enter the choice
9
enter item
1
search unsuccessfull
enter the choice
3
the node deleted is 6
enter the choice
3
the node deleted is 12
enter the choice
4
```



LAB 10-

Write a program

a]To construct a binary search tree b]to traverse the tree using methods i.e. inorder, preorder, postorder c]to display the elements of the tree

```
#include<stdio.h>

#include<stdlib.h>

struct node

 {

 int info;

 struct node *rlink;

 struct node *llink;

};
```

```
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
 printf("mem full\n");
 exit(0);
 }
 return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert(NODE root,int item)
{
NODE temp,cur,prev;
temp=getnode();
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
if(root==NULL)
```

```
 return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
 prev->llink=temp;
else
 prev->rlink=temp;
return root;
}
void display(NODE root,int i)
{
int j;
if(root!=NULL)
 {
 display(root->rlink,i+1);
 for(j=0;j<i;j++)
 printf(" ");
 printf("%d\n",root->info);
 display(root->llink,i+1);
```

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

```
 inorder(root->llink);
 printf("%d\n",root->info);
 inorder(root->rlink);
 }
 }
int main()
{
int item,choice;
NODE root=NULL;
do
{
printf("\n1.insert\n2.display\n3.preorder\n4.postorder\n5.inorder\n6.exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
 case 1:printf("enter the item\n");
 scanf("%d",&item);
 root=insert(root,item);
 break;
 case 2:display(root,0);
 break;
 case 3:preorder(root);
 break;
```

```
case 4:postorder(root);
break;
case 5:inorder(root);
break;
case 6: break;
default:exit(0);
break;
}
}while(choice!=6);
return 0;
}
```

Output-

```
1.insert
2.display
3.preorder
4.postorder
5.inorder
6.exit
enter the choice
1
enter the item
100

1.insert
2.display
3.preorder
4.postorder
5.inorder
6.exit
enter the choice
1
enter the item
20

1.insert
2.display
3.preorder
4.postorder
5.inorder
6.exit
enter the choice
1
```

```
enter the choice
1
enter the item
30

1.insert
2.display
3.preorder
4.postorder
5.inorder
6.exit
enter the choice
1
enter the item
10

1.insert
2.display
3.preorder
4.postorder
5.inorder
6.exit
enter the choice
1
enter the item
200

1.insert
2.display
3.preorder
4.postorder
```

```
4.postorder
5.inorder
6.exit
enter the choice
1
enter the item
150

1.insert
2.display
3.preorder
4.postorder
5.inorder
6.exit
enter the choice
1
enter the item
300

1.insert
2.display
3.preorder
4.postorder
5.inorder
6.exit
enter the choice
2
    300
  200
    150
100
```

```
enter the choice
2
    300
  200
    150
100
    30
  20
    10

1.insert
2.display
3.preorder
4.postorder
5.inorder
6.exit
enter the choice
3
100
20
10
30
200
150
300

1.insert
2.display
3.preorder
4.postorder
5.inorder
```

```
2.display
3.preorder
4.postorder
5.inorder
6.exit
enter the choice
4
10
30
20
150
300
200
100

1.insert
2.display
3.preorder
4.postorder
5.inorder
6.exit
enter the choice
5
10
20
30
100
150
200
300
```

```
200
100

1.insert
2.display
3.preorder
4.postorder
5.inorder
6.exit
enter the choice
5
10
20
30
100
150
200
300

1.insert
2.display
3.preorder
4.postorder
5.inorder
6.exit
enter the choice
6


...Program finished with exit code 0
Press ENTER to exit console.
```