Lab-2 STACK IMPLEMENTATION:

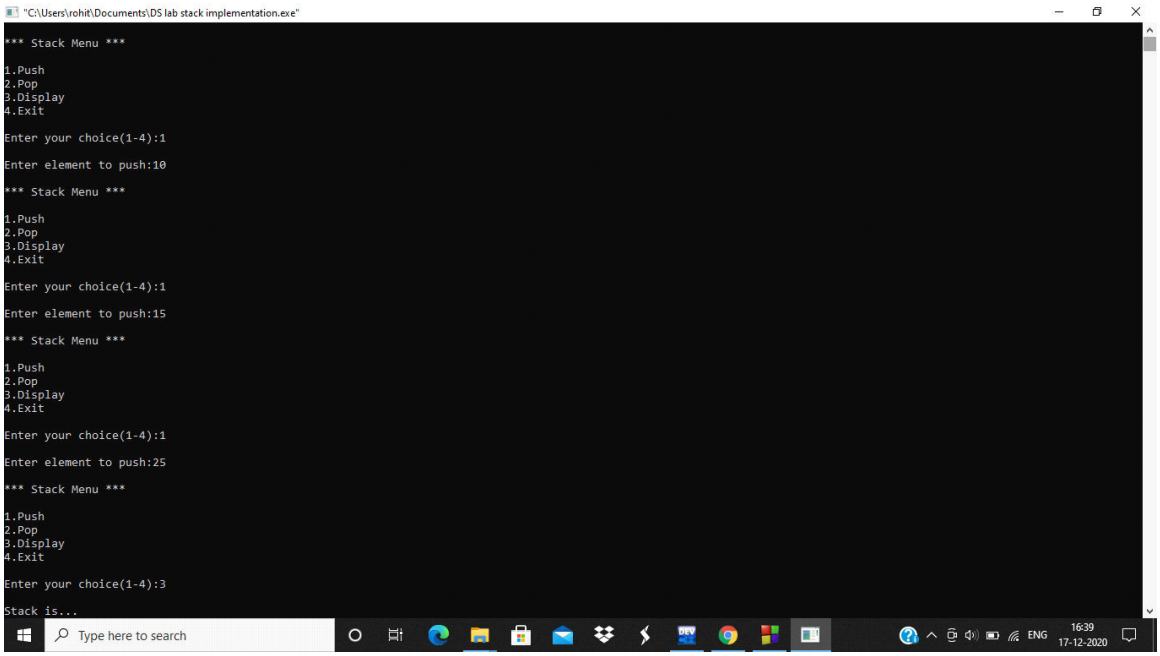
```
#include<stdio.h>
#include<process.h>
#include<stdlib.h>

#define MAX 5

int top=-1,stack[MAX];
void push();
void pop();
void display();

void main()
{
        int ch;

        while(1)
        {
                printf("\n*** Stack Menu ***");
                printf("\n\n1.Push\n2.Pop\n3.Display\n4.Exit");
                printf("\n\nEnter your choice(1-4):");
                scanf("%d",&ch);

                switch(ch)
                {
```

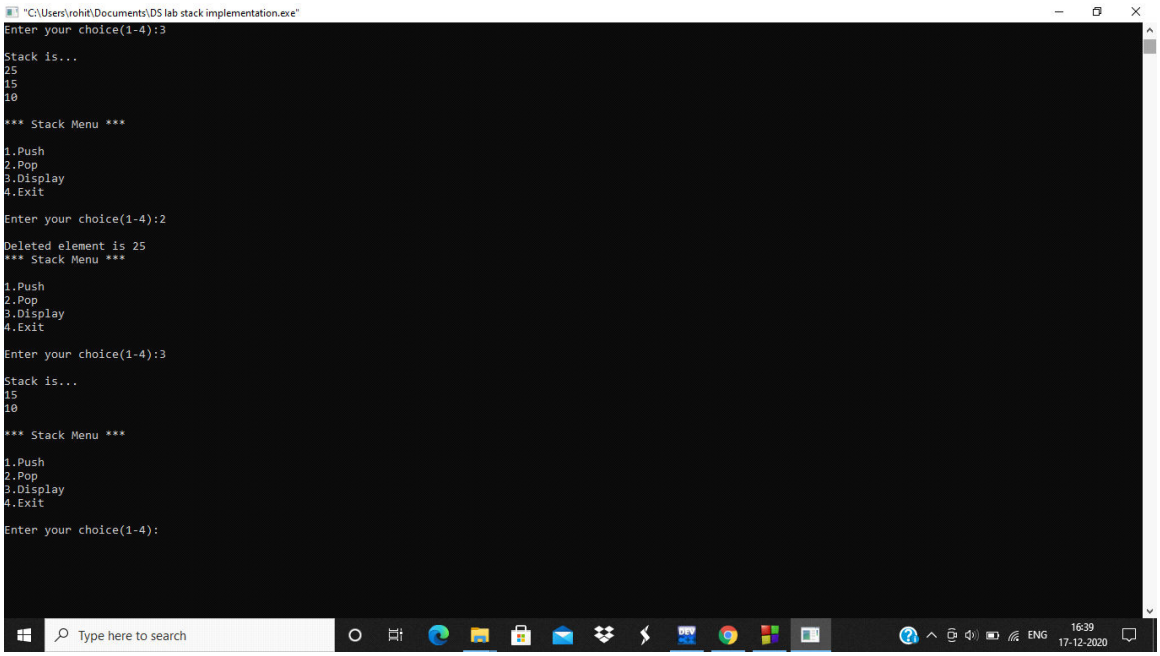
```
                case 1: push();
                        break;
                case 2: pop();
                        break;
                case 3: display();
                        break;
                case 4: exit(0);

                default: printf("\nWrong Choice!!");
            }
        }
}

void push()
{
        int val;

        if(top==MAX-1)
        {
            printf("\nStack is full!!");
        }
        else
        {
            printf("\nEnter element to push:");
            scanf("%d",&val);
            top=top+1;
```

```
                stack[top]=val;
        }
}

void pop()
{
        if(top==-1)
        {
                printf("\nStack is empty!!");
        }
        else
        {
                printf("\nDeleted element is %d",stack[top]);
                top=top-1;
        }
}

void display()
{
        int i;

        if(top==-1)
        {
                printf("\nStack is empty!!");
        }
        else
```

```
        {
                printf("\nStack is...\n");
                for(i=top;i>=0;--i)
                        printf("%d\n",stack[i]);
        }
}
```

OUTPUT:





Lab-3 INFIX TO POSTFIX:

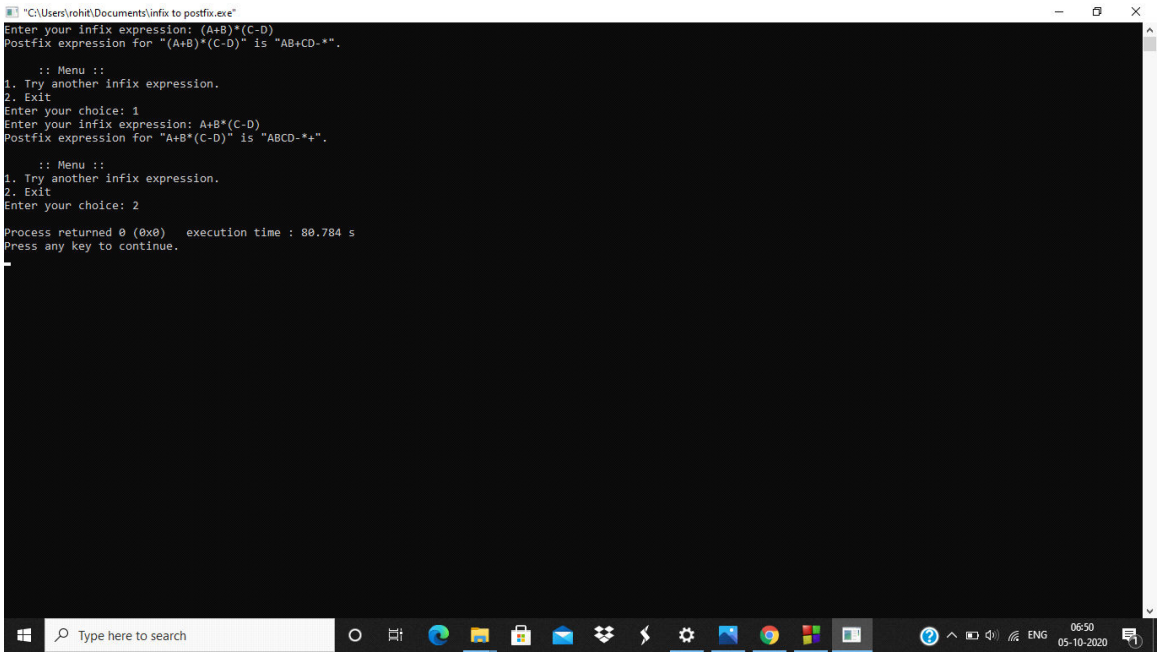
```
#include <stdio.h>

#define N 100


int stack[N];


int top = -1;


void push(int item){
        if(top==N-1)
                printf("Stack overflow!\n");
        else
                stack[++top] = item;
```

```
}

int pop(){
	if(top==-1)
		printf("Stack underflow!\n");
	else
		return stack[top--];

}

int priority(char op){
	switch(op){
		case '*':	return 2;
				break;
		case '/':	return 2;
				break;
		case '+':	return 1;
				break;
		case '-':	return 1;
				break;
		case '(':	return 0;
				break;
	}
}

int main()
```

```
{
	char s[50];
	char t[50];
	int l;
	int choice = 1;

	do{
		l = 0;
		printf("Enter your infix expression: ");
		scanf("%s", s);

		for(int i=0; s[i]!='\0'; i++){
			switch(s[i]){
				case '(':	push('(');
						break;
				case ')':	while(stack[top]!='('){
							t[l++] = pop();
						}
						pop();
						break;
				case '*':
				case '/':
				case '+':
				case '-':	while(top!=-1 && priority(stack[top])>=priority(s[i])){
							t[l++] = pop();
						}
```

```
                                        push(s[i]);
                                        break;
                        default: t[l++] = s[i];
                }
        }

        while(top!=-1){
                t[l++] = pop();
        }

        t[l] = '\0';

        printf("Postfix expression for \"%s\" is \"%s\".\n", s, t);

        printf("\n          :: Menu ::          \n1. Try another infix expression.\n2. Exit\nEnter your choice:
");

        scanf("%d", &choice);
    }while(choice!=2);

    return 0;
}
```

OUTPUT:



Lab-4 QUEUE:

```
#include<stdio.h>
#include<math.h>
#include<string.h>
#include<stdlib.h>
#define max 5
void insert_right();
void insert_left();
void delete_right();
void delete_left();
void display();
void input();
void output();
```

```
int q[max];
int left=-1;
int right=-1;
int main()
{
	int option;
	printf("1: INPUT RESTRICTED\n");
	printf("2: OUTPUT RESTRICTED\n");
	printf("3: EXIT\n");
	printf("Enter Your Choice \n");
	scanf("%d",&option);
	switch(option)
	{

	case 1:
		input();
		break;
	case 2:
		output();
		break;
	case 3:
		exit(0);
	}

}
void input()
```

```
{
    int option;
    do
    {
        printf("1:Insert Right\n");
        printf("2:Delete Right\n");
        printf("3:Delete Left\n");
        printf("4:Display\n");
        printf("5:Exit\n");
        printf("Enter your Option\n");
        scanf("%d",&option);
        switch(option)
        {

        case 1:
            insert_right();
            break;
        case 2:
            delete_right();
            break;
        case 3:
            delete_left();
            break;
        case 4:
            display();
            break;
```

```
            case 5:
                exit(0);
            }
        }while(option!=5);
}

void output()
{
        int option;
        do
        {
            printf("1:Delete Left\n");
            printf("2:Insert Right\n");
            printf("3:Insert Left\n");
            printf("4:Display\n");
            printf("5:EXIT\n");
            printf("Enter Your Option\n");
            scanf("%d",&option);
            switch(option)
            {
            case 1:
                delete_left();
                break;
            case 2:
                insert_right();
                break;
```

```
            case 3:
                insert_left();
                break;
            case 4:
                display();
                break;
            case 5:
                exit(0);
            }
        }while(option!=5);
}

void insert_right()
{
        int num;
        printf("Enter The Number To Be Inserted \n");
        scanf("%d",&num);
        if((left==0&&right==max-1)||left==right+1)
        {
            printf("Queue Overflow\n");
            exit(0);
        }
        if(left==-1&&right==-1)
        {
            left=0;
            right=0;
```

```
        }
        else
        {
                if(right==max-1)
                        right=0;
                else
                        right++;
        }
        q[right]=num;
}
void insert_left()
{
        int num;
        printf("Enter The Number To Be Inserted \n");
        scanf("%d",&num);
        if((left==0&&right==max-1)||left==right+1)
        {
                printf("Queue Overflow\n");
                exit(0);
        }
        if(left==-1&&right==-1)
        {
                left=0;
                right=0;
        }
        else
```

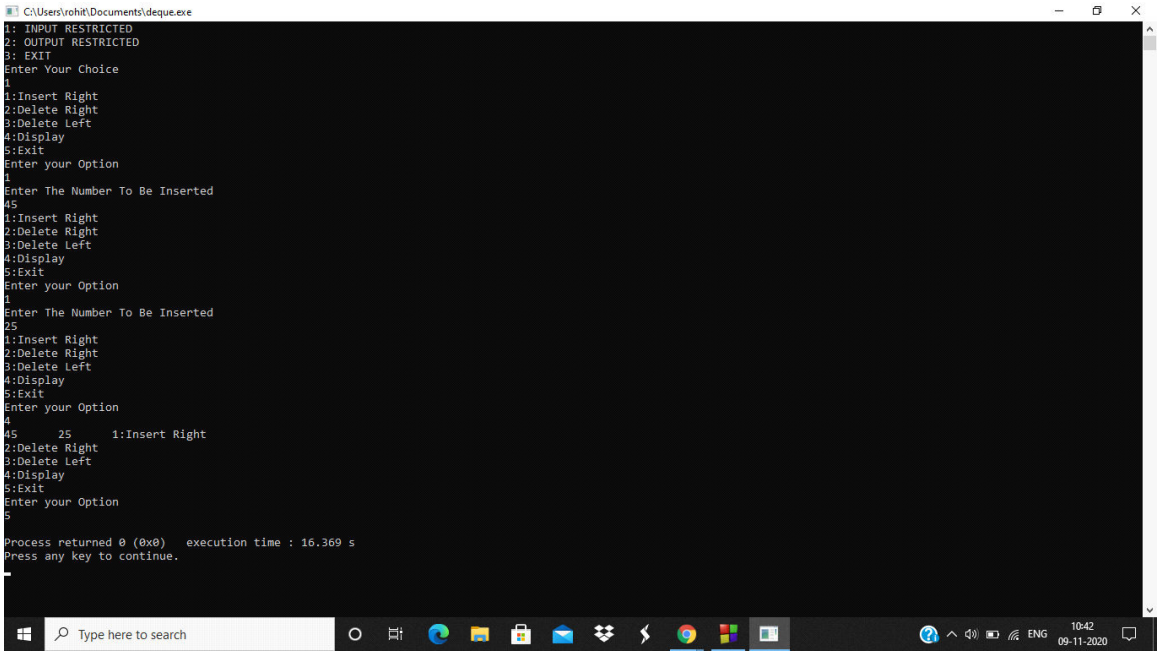
```
    {
        if(left==0)
            left=max-1;
        else
            left--;
    }
    q[left]=num;
}
void delete_left()
{
    if(left==-1)
    {
        printf("Queue Underflow\n");
    }
    printf("Element Deleted is %d\n",q[left]);
    if(left==right)
    {
        left=-1;
        right=-1;
    }
    else
    {
        if(left==max-1)
            left=0;
        else
            left++;
```

```
        }
}
void delete_right()
{
        if(left==-1)
        {
                printf("Queue Underflow\n");
        }
        printf("Element Deleted is %d\n",q[right]);
        if(left==right)
        {
                left=-1;
                right=-1;
        }
        else
        {
                if(right==0)
                        right=max-1;
                else
                        right--;
        }
}
void display()
{
        int front=left;
        int rear=right;
```

```
if(left==-1)
{
	printf("Queue Is Empty\n");
	exit(0);
}
if(front<=rear)
{
	while(front<=rear)
	{
		printf("%d\t",q[front]);
		front++;
	}
}
else
{
	while(front<=max-1)
	{
		printf("%d\t",q[front]);
		front++;
	}
	front=0;
	while(front<=rear)
	{
		printf("%d\t",q[front]);
		front++;
	}
```

```
    }
}
```

OUTPUT:



LAB-5 CIRCULAR QUEUE:

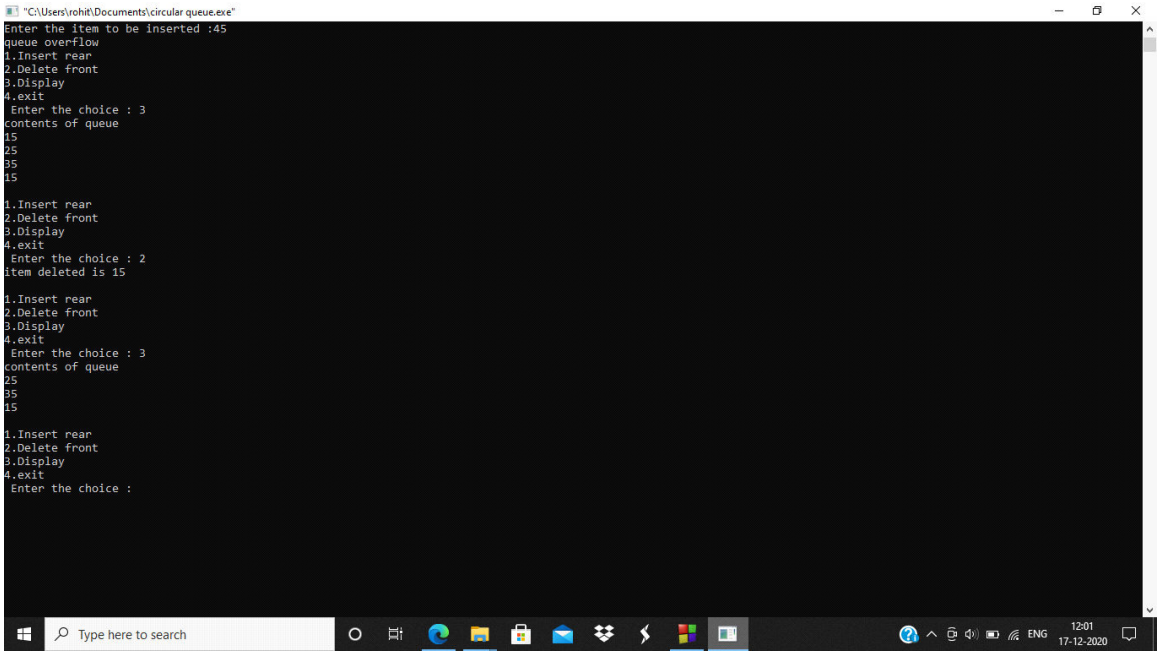
```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
#define que_size 3
int item,front=0,rear=-1,q[que_size],count=0;
void insertrear()
{
        if(count==que_size)
        {
```

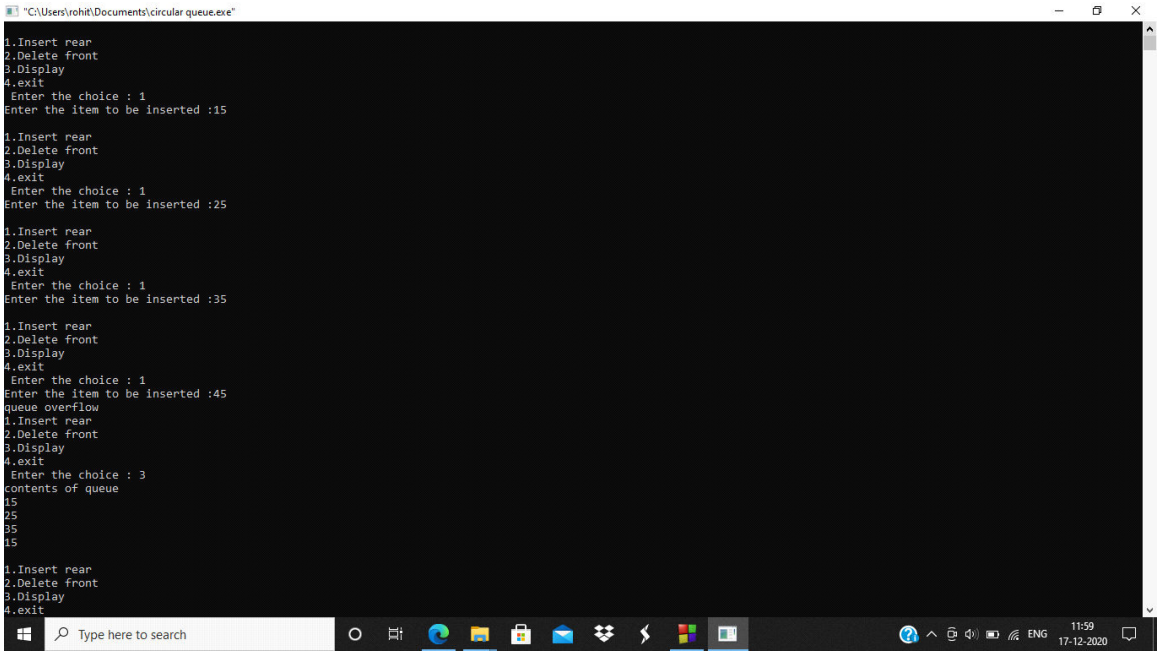
```
                printf("queue overflow");
                return;
        }
        rear=(rear+1)%que_size;
        q[rear]=item;
        count++;
}
int deletefront()
{
        if(count==0) return -1;
        item = q[front];
        front=(front+1)%que_size;
        count=count-1;
        return 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();
}
```

OUTPUT:





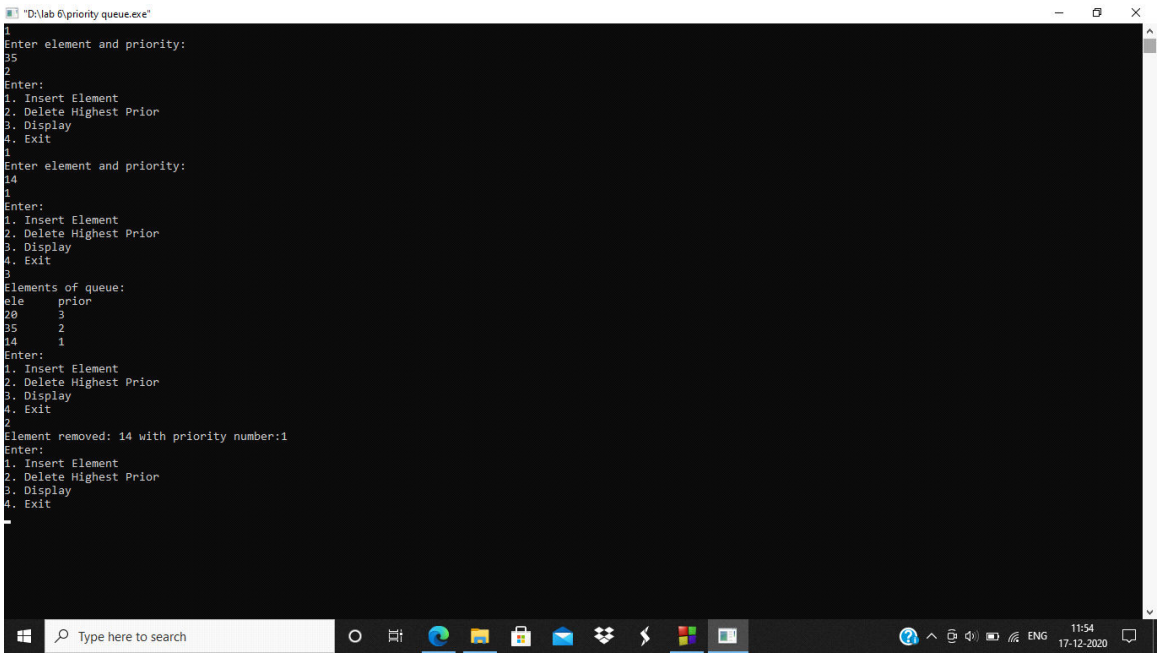
LAB-6 PRIORITY QUEUE:

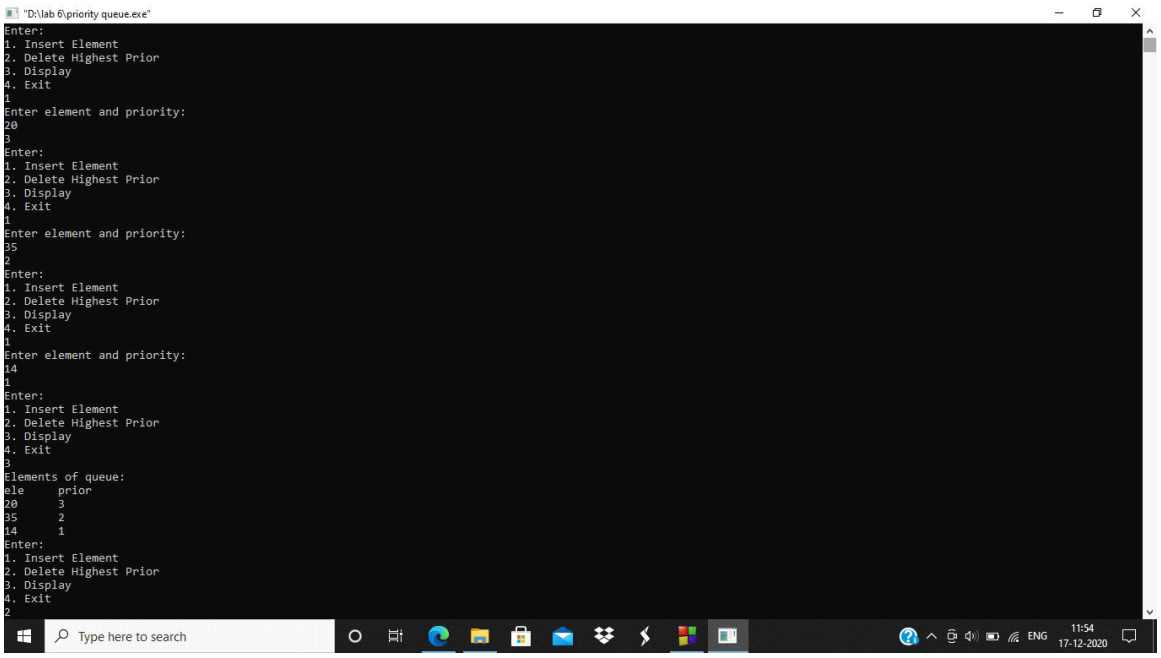
```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#define quesize 10
int item, p, rear=-1, q[quesize][2];
void insrear(){
  if(rear<quesize){
  q[++rear][0]=item;
  q[rear][1]=p;
  }
  else
  printf("Queue overflow\n");
}
void remove_small(){
  int min=INT_MAX;
  int t;
  for(int i=0;i<=rear;i++){
  if(q[i][1]<min){
  min=q[i][1];
  t=i;
  }
  }
  if(min!=INT_MAX){
  printf("Element removed: %d with priority number:%d\n",q[t][0],min);
  q[t][1]=INT_MAX;
  }
```

```
  else
  printf("Queue Underflow\n");
}
void display(){
  printf("Elements of queue:\nele\tprior\n");
  for(int i=0;i<=rear;i++){
  if(q[i][1]!=INT_MAX)
  printf("%d\t%d\n",q[i][0],q[i][1]);
  }
}
int main(){
  int choice;
  for(;;){
  printf("Enter:\n1. Insert Element\n2. Delete Highest Prior\n3. Display\n4. Exit\n");
  scanf("%d",&choice);
  switch (choice){
  case 1: printf("Enter element and priority:\n");
  scanf("%d%d", &item,&p);
  insrear();
  break;
  case 2: remove_small();
  break;
  case 3: display();
  break;
  case 4: exit(0);
  default: printf("Wrong choice\n");
  }
  }
```

```
  return 0;
}
```

OUTPUT:





LAB-7 SINGILY LINKED 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;
NODE prev,cur;
int count;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL && pos==1)
return temp;
if(first==NULL)
{
printf("invalid pos\n");
```

```
return first;
}
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("IP\n");
return first;
}
NODE delete_pos(int pos, NODE first){
```

```
 if (first == NULL){
 printf("List empty\n");
 return first;
 }
 NODE temp= first;
 if (pos==1)
 {
 first = temp->link;
 free(temp);
 return first;
 }
 NODE prev;

 for (int i=1; temp!=NULL && i<pos; i++){
 prev=temp;
 temp = temp->link;
 }
 if (temp == NULL || temp->link == NULL){
 printf("Invalid position\n");
 return NULL;
 }
 prev->link=temp->link;
 printf("Element deleted %d\n",temp->info);
 free(temp);
 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);
  }
}
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;
}
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 order_list(NODE first)
{
  int swapped, i;
  NODE ptr1,lptr=NULL;
  if (first == NULL)
  return first;

  do
  {
  swapped = 0;
  ptr1 = first;

  while (ptr1->link != lptr)
  {
  if (ptr1->info > ptr1->link->info)
  {
```

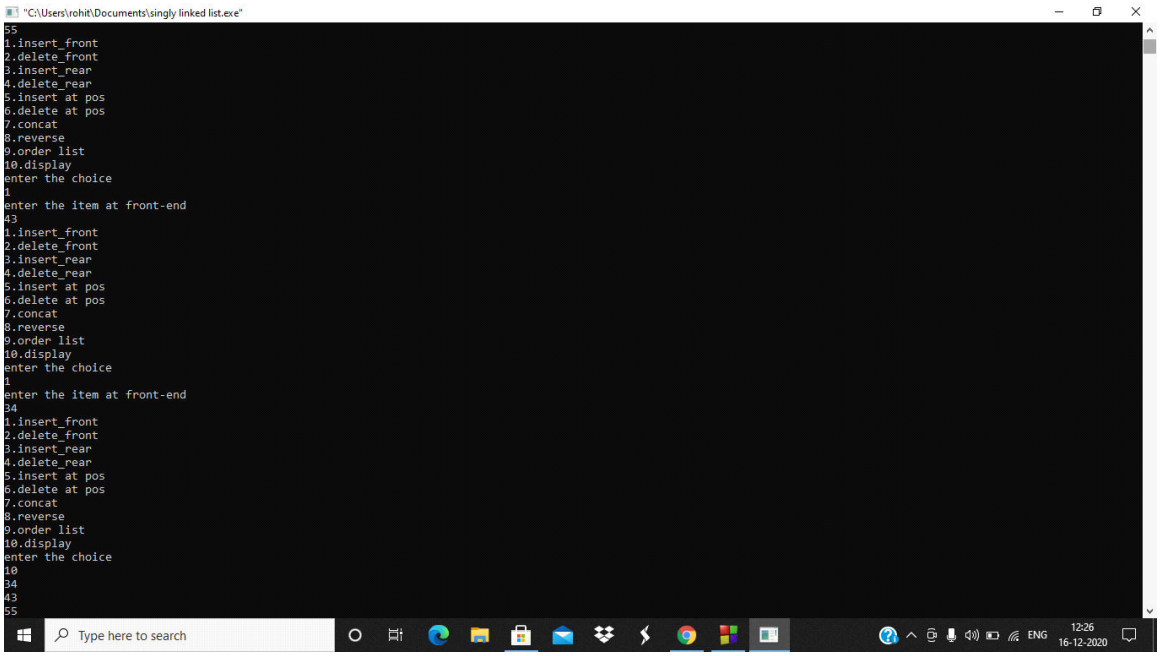
```
  int temp = ptr1->info;
  ptr1->info = ptr1->link->info;
  ptr1->link->info = temp;
  swapped = 1;
  }
  ptr1 = ptr1->link;
  }
  lptr = ptr1;
  }
  while (swapped);
  return first;
}
void main()
{
int item,choice,pos,i,n;
NODE a,b;
NODE first=NULL;
for(;;)
{
printf("1.insert_front\n2.delete_front\n3.insert_rear\n4.delete_rear\n5.insert at pos\n6.delete at
pos\n7.concat\n8.reverse\n9.order list\n10.display\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
  case 1:printf("enter the item at front-end\n");
```

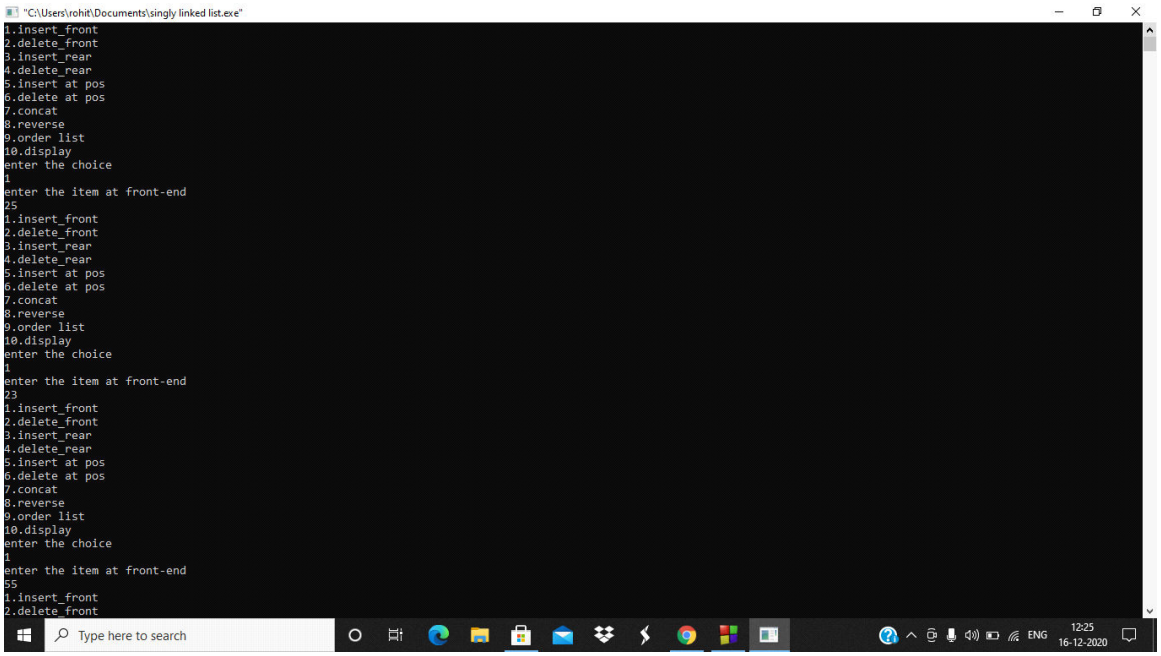
```
scanf("%d",&item);
first=insert_front(first,item);
break;
case 2:first=delete_front(first);
break;
case 3:printf("enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 4:first=delete_rear(first);
break;
case 5:
printf("Enter item\n");
scanf("%d",&item);
printf("enter the position\n");
scanf("%d",&pos);
first=insert_pos(item,pos,first);
break;
case 6:
printf("Enter posititon of deletion\n");
scanf("%d",&pos);
first=delete_pos(pos,first);
break;
case 7:
printf("enter the no of nodes in 1\n");
scanf("%d",&n);
```

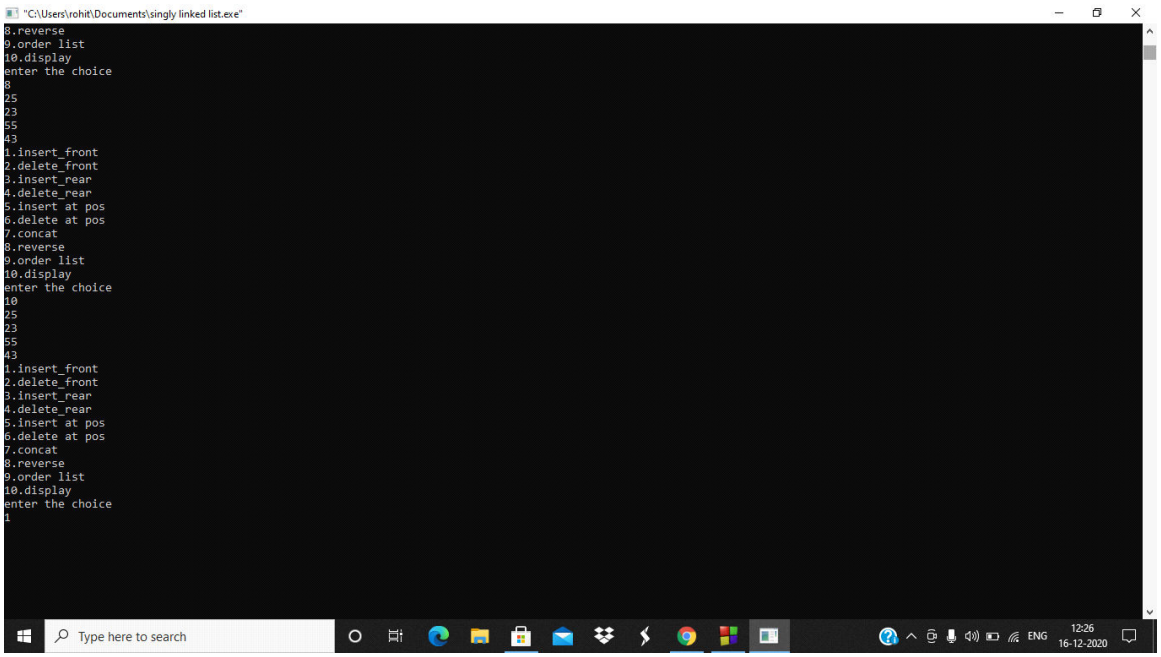
```
a=NULL;
for(i=0;i<n;i++)
{
printf("enter the item\n");
scanf("%d",&item);
a=insert_rear(a,item);
}
printf("enter the no of nodes in 2\n");
scanf("%d",&n);
b=NULL;
for(i=0;i<n;i++)
{
printf("enter the item\n");
scanf("%d",&item);
b=insert_rear(b,item);
}
a=concat(a,b);
display(a);
break;
case 8:
first=reverse(first);
display(first);
break;
case 9:
first=order_list(first);
break;
```

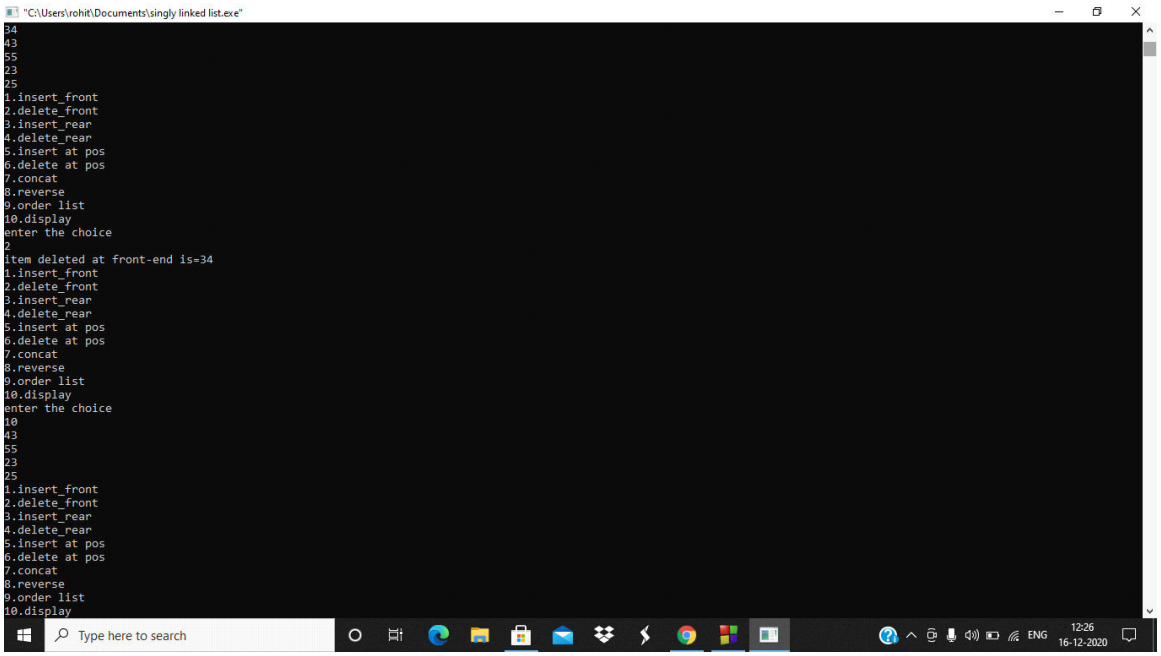
```
  case 10:display(first);
  break;
default:exit(0);
  break;
}
}
}
```

OUTPUT:




"C:\Users\rohit\Documents\singly linked list.exe"
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
1
enter the item at front-end
25
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
1
enter the item at front-end
23
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
1
enter the item at front-end
55
1.insert_front
2.delete_front
Type here to search
ENG
12:25
16-12-2020


"C:\Users\rohit\Documents\singly linked list.exe"
8.reverse
9.order list
10.display
enter the choice
8
25
23
55
43
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
10
25
23
55
43
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
1
Type here to search
ENG
12:26
16-12-2020


"C:\Users\rohit\Documents\singly linked list.exe"
34
43
55
23
25
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
2
item deleted at front-end is=34
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
enter the choice
10
43
55
23
25
1.insert_front
2.delete_front
3.insert_rear
4.delete_rear
5.insert at pos
6.delete at pos
7.concat
8.reverse
9.order list
10.display
Type here to search
ENG
12:26
16-12-2020