NAME - NOOR FATHIMA ARFA

USN - 1BM19CS108

SUBJECT - DATA STRUCTURE

ACADEMIC YEAR - 2020-21

LAB PROGRAM

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

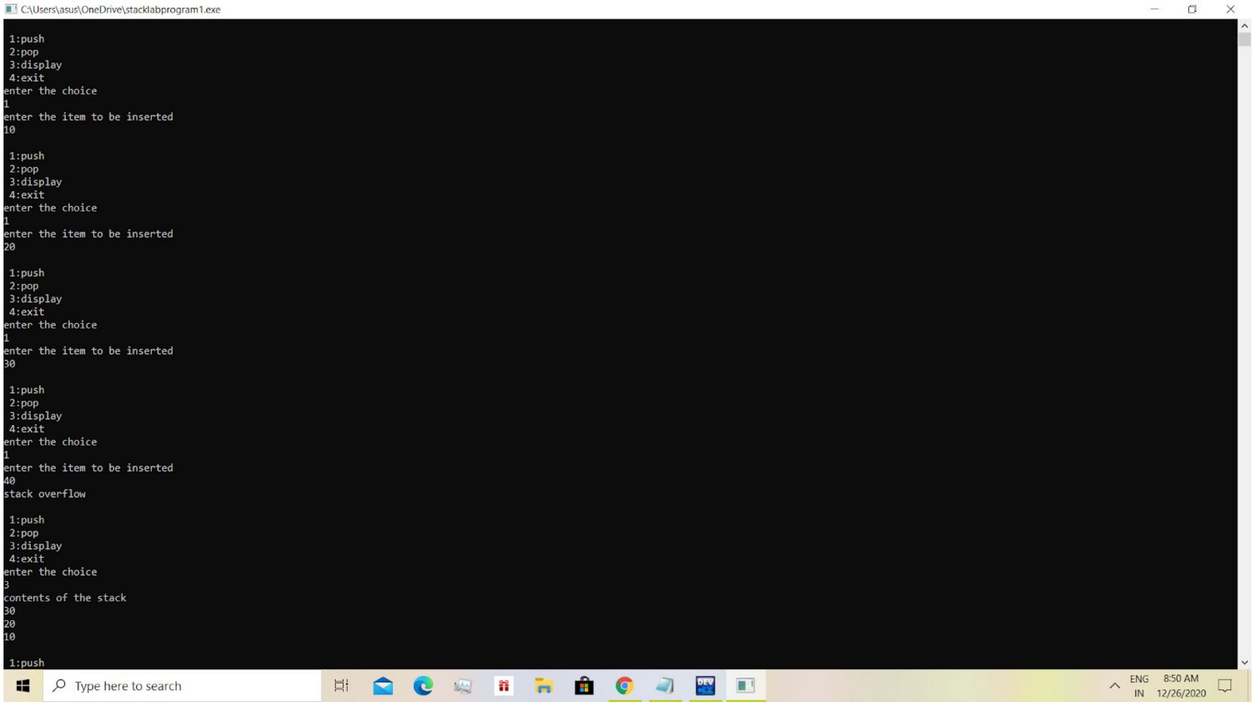
a) Push

b) Pop

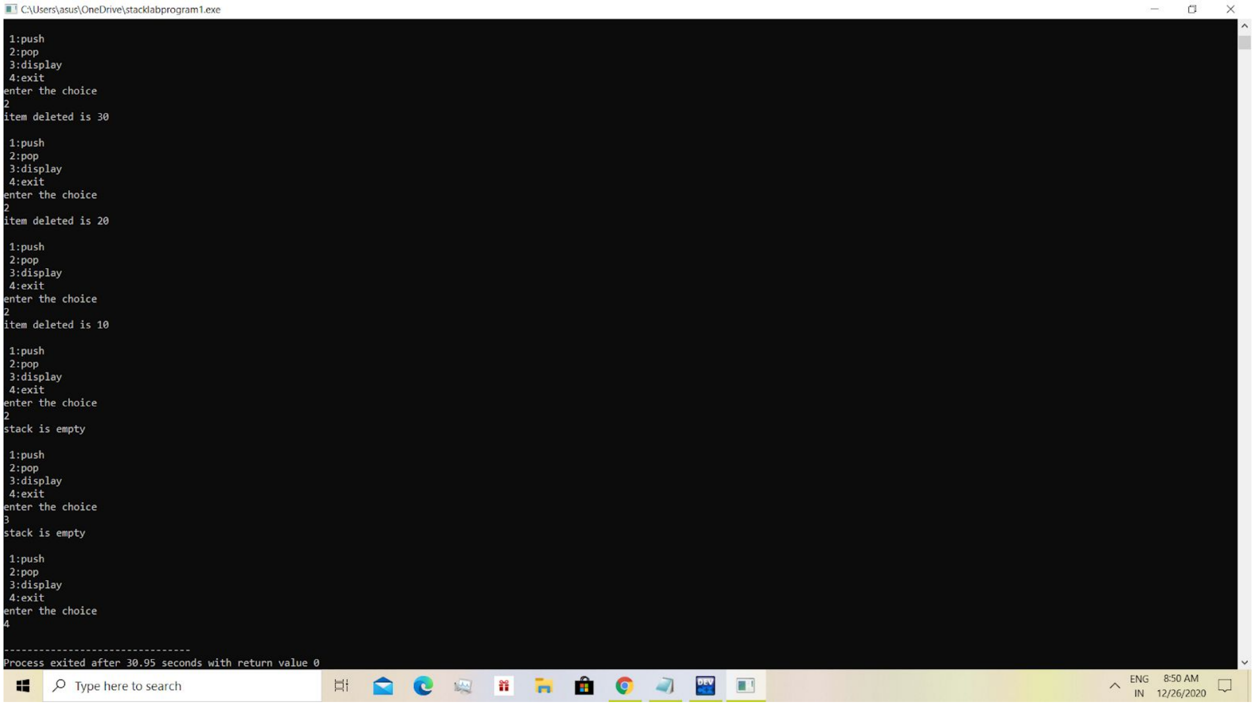
c) Display

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

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#define STACK_SIZE 3
int top=-1;
int s[10];
int 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\n");
for(i=top; i>=0; i--)
{
printf("%d\n",s[i]);
}
}
void main()
{
int item_deleted;
int choice;

for(;;)
{
printf("\n 1:push\n 2:pop\n 3:display\n 4: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);
push();
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:exit(0);
}
}

}
```



```
 1:push
 2:pop
 3:display
 4:exit
enter the choice
1
enter the item to be inserted
10

 1:push
 2:pop
 3:display
 4:exit
enter the choice
1
enter the item to be inserted
20

 1:push
 2:pop
 3:display
 4:exit
enter the choice
1
enter the item to be inserted
30

 1:push
 2:pop
 3:display
 4:exit
enter the choice
1
enter the item to be inserted
40
stack overflow

 1:push
 2:pop
 3:display
 4:exit
enter the choice
3
contents of the stack
30
20
10

 1:push
```



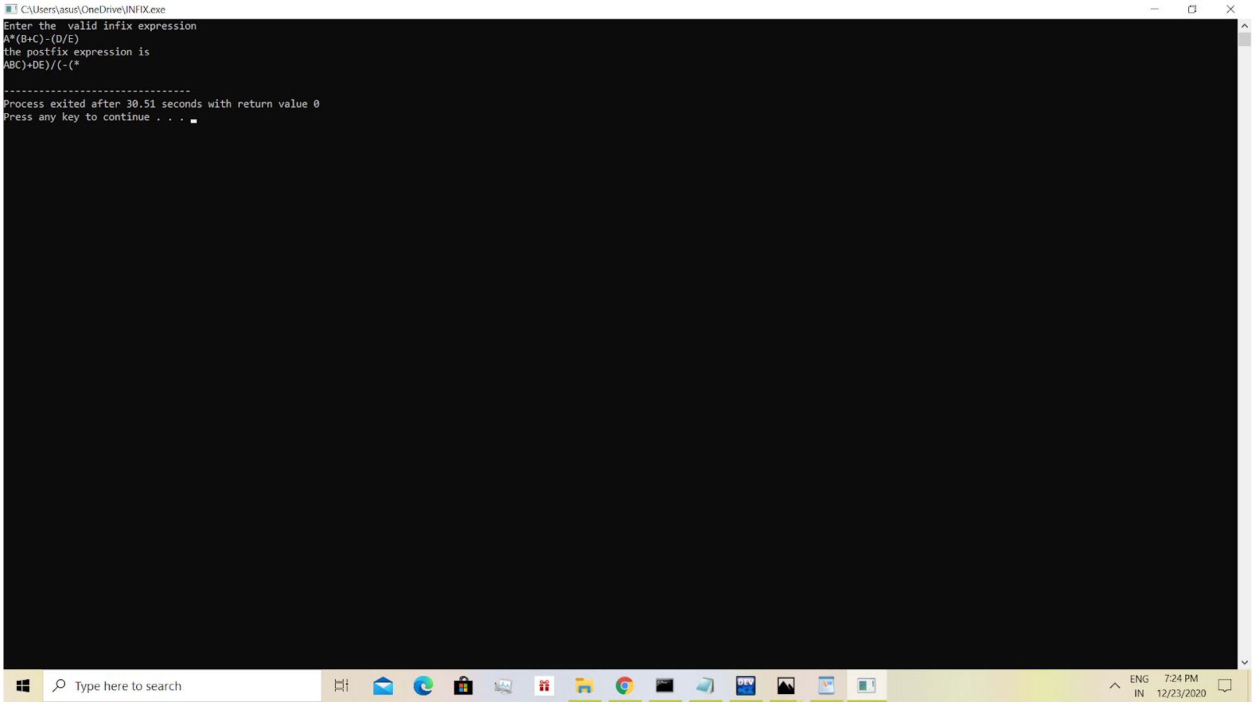
```
 1:push
 2:pop
 3:display
 4:exit
enter the choice
2
item deleted is 30

 1:push
 2:pop
 3:display
 4:exit
enter the choice
2
item deleted is 20

 1:push
 2:pop
 3:display
 4:exit
enter the choice
2
item deleted is 10

 1:push
 2:pop
 3:display
 4:exit
enter the choice
2
stack is empty

 1:push
 2:pop
 3:display
 4:exit
enter the choice
3
stack is empty

 1:push
 2:pop
 3:display
 4:exit
enter the choice
4

--------------------------------
Process exited after 30.95 seconds with return value 0
```

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<stdlib.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 the  valid infix expression\n");
          scanf("%s",infix);
          infix_postfix(infix,postfix);
          printf("the postfix expression is\n");
```

```
    printf("%s\n",postfix);
  }
```



```
Enter the  valid infix expression
A*(B+C)-(D/E)
the postfix expression is
ABC)+DE)/(-(*

--------------------------------
Process exited after 30.51 seconds with return value 0
Press any key to continue . . .
```

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

a) Insert

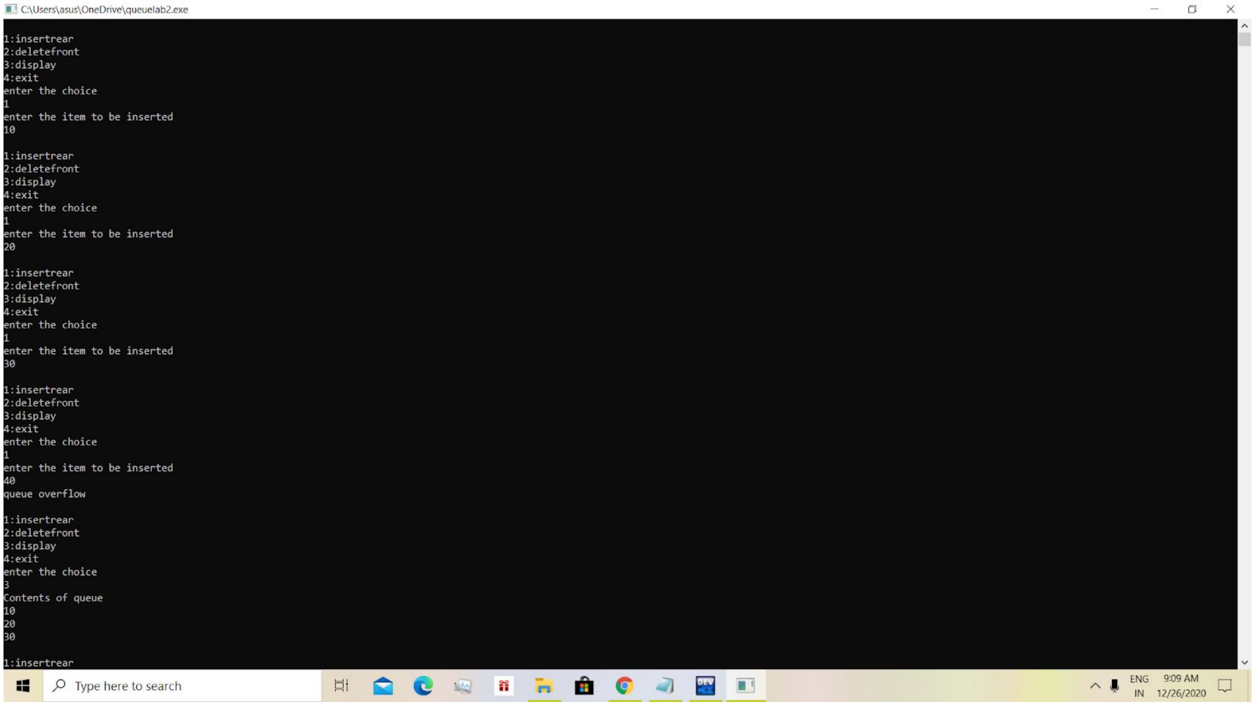
b) Delete

c) Display

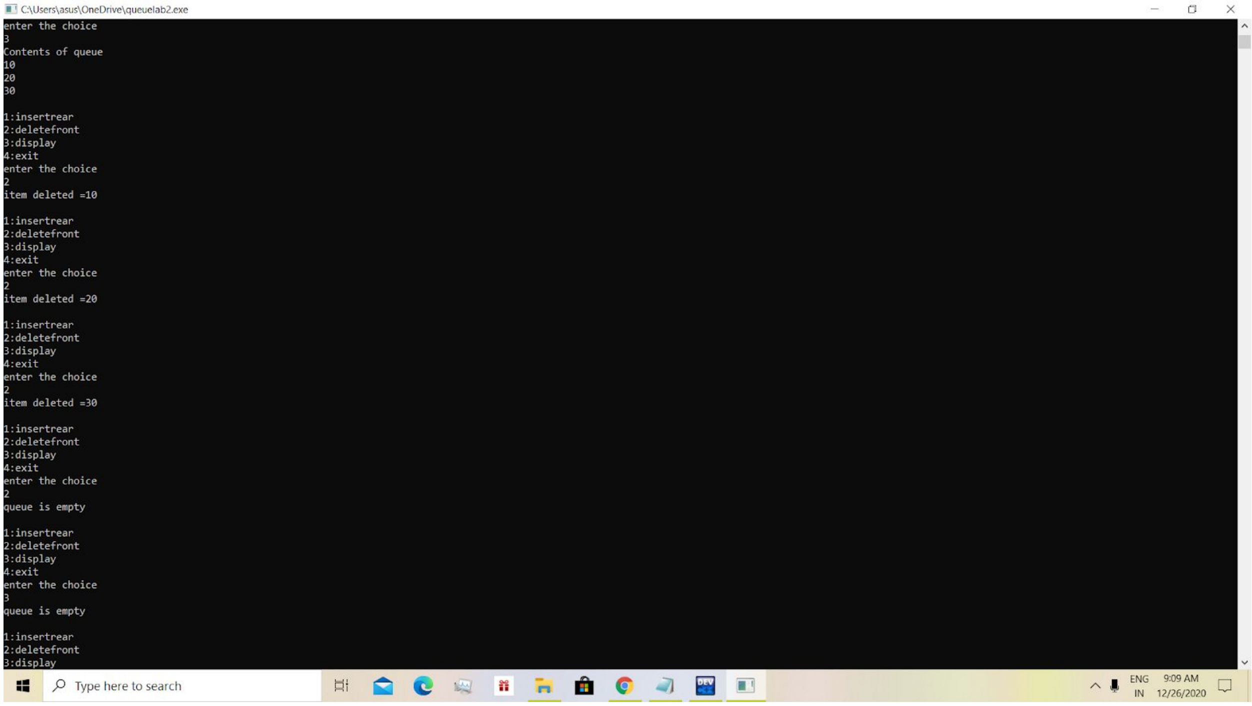
The program should print appropriate messages for queue empty and queue empty conditions

```
#include<stdio.h>
#include<conio.h>
#include<process.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)
 return -1;
return q[front++];
}
void displayQ()
{
int i;
if(front>rear)
{
printf("queue is empty\n");
return;
}
printf("Contents of queue \n");
for(i=front;i<=rear;i++)
{
printf("%d\n",q[i]);
}
}
void main()
{
int choice;
for(;;)
{
printf("\n1:insertrear\n2:deletefront\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 =%d\n",item);
break;
case 3:displayQ();
break;
default:exit(0);
}
}
}
```



```
1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
10

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
20

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
30

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
40
queue overflow

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
3
Contents of queue
10
20
30

1:insertrear
```



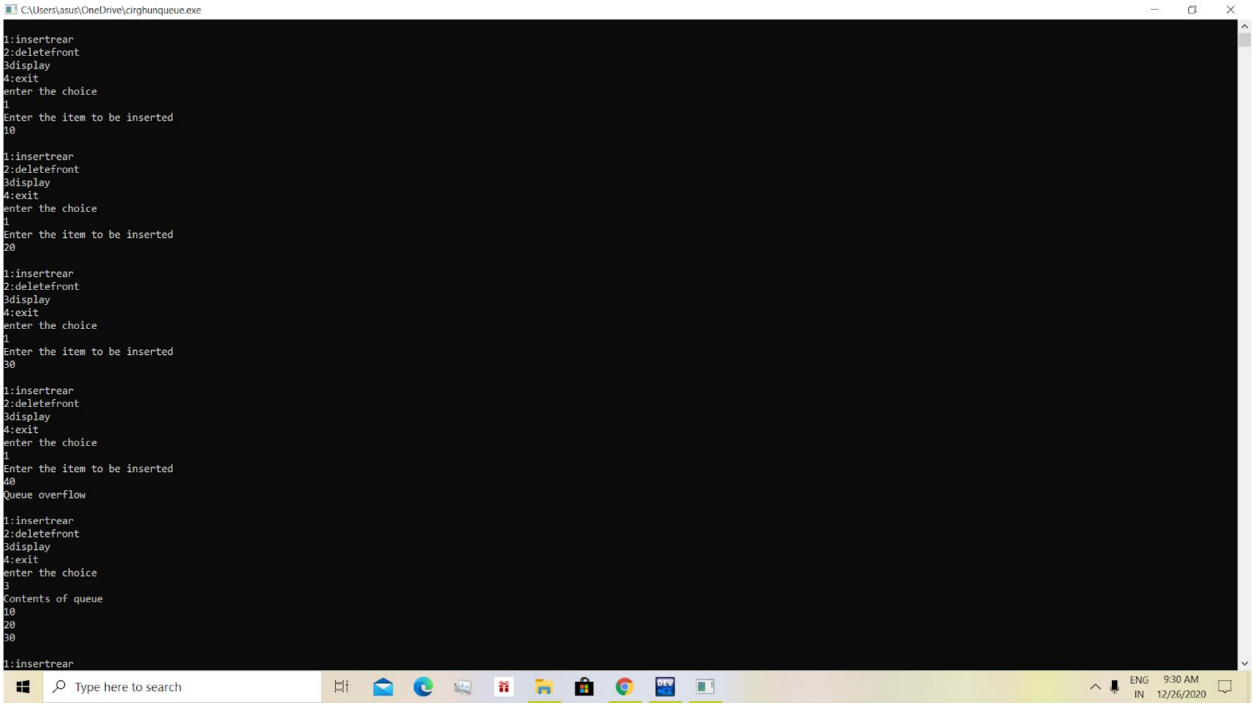
```
enter the choice
3
Contents of queue
10
20
30

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =10

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =20

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =30

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
queue is empty

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
3
queue is empty

1:insertrear
2:deletefront
3:display
```

4)WAP to simulate the working of a Circular queue of integers using an array. Provide the following operations

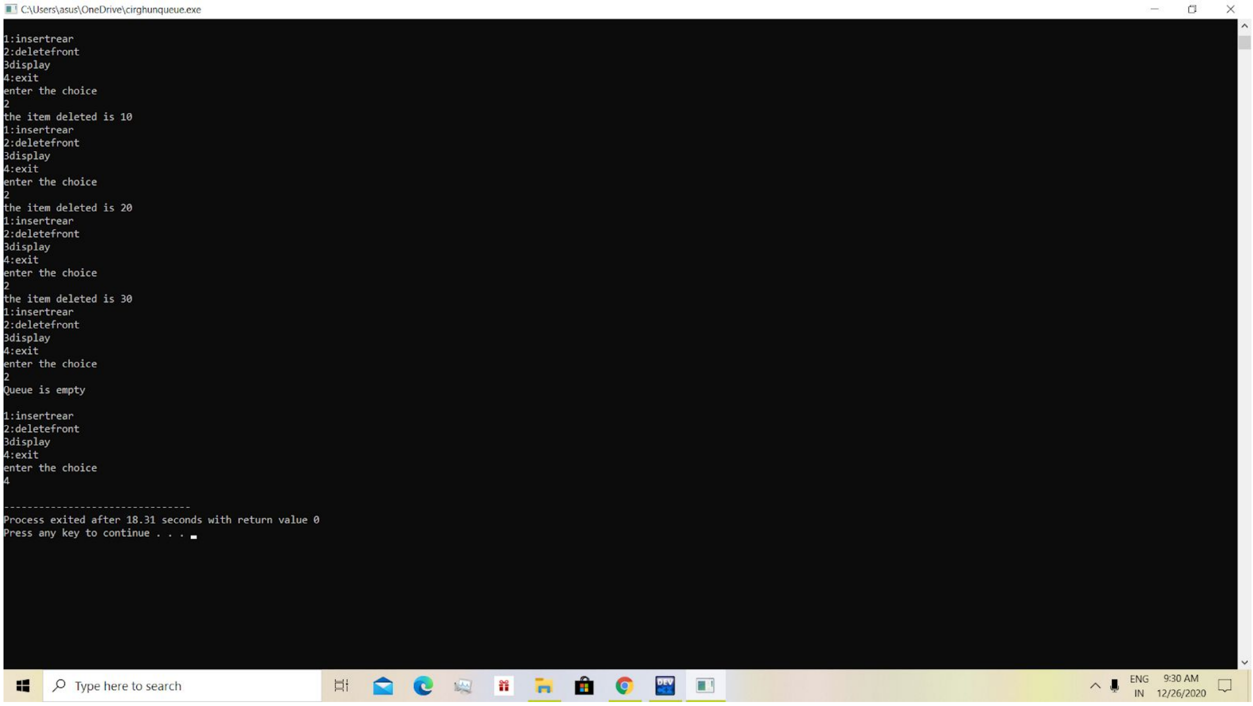
a) Insert
b) Delete
c) Display
The program should print appropriate messages for queue empty and queue overflow conditions

```
#include<stdio.h>
#include<conio.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\n");
                return;
        }

        else
        rear=(rear+1)%QUE_SIZE;
        q[rear]=item;
        count++;
}
int deletefront()
{
        if(count==0)
        return -1;
        else
        item=q[front];
        front=(front+1)%QUE_SIZE;
        count=count-1;
        return item;
}
void display()
{
        int i;
        if(count==0)
        {
                printf("Queue iis empty");
                return ;
        }
        else
```

```
    printf("Contents of queue \n");
    for(i=1;i<=count;i++)
    {
        printf("%d\n",q[front]);
        front=(front+1)%QUE_SIZE;
    }

}
void main()
{
    int choice;
    for(;;)
    {
        printf("\n1:insertrear\n2:deletefront\n3display\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("the item deleted is %d",item);
                break;
            case 3:
                display();
              break;
          case 4:
            exit(0);

        }
    }
}
```



```
1:insertrear
2:deletefront
3display
4:exit
enter the choice
1
Enter the item to be inserted
10

1:insertrear
2:deletefront
3display
4:exit
enter the choice
1
Enter the item to be inserted
20

1:insertrear
2:deletefront
3display
4:exit
enter the choice
1
Enter the item to be inserted
30

1:insertrear
2:deletefront
3display
4:exit
enter the choice
1
Enter the item to be inserted
40
Queue overflow

1:insertrear
2:deletefront
3display
4:exit
enter the choice
3
Contents of queue
10
20
30

1:insertrear
```



```
1:insertrear
2:deletefront
3display
4:exit
enter the choice
2
the item deleted is 10
1:insertrear
2:deletefront
3display
4:exit
enter the choice
2
the item deleted is 20
1:insertrear
2:deletefront
3display
4:exit
enter the choice
2
the item deleted is 30
1:insertrear
2:deletefront
3display
4:exit
enter the choice
2
Queue is empty

1:insertrear
2:deletefront
3display
4:exit
enter the choice
4

--------------------------------
Process exited after 18.31 seconds with return value 0
Press any key to continue . . .
```

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

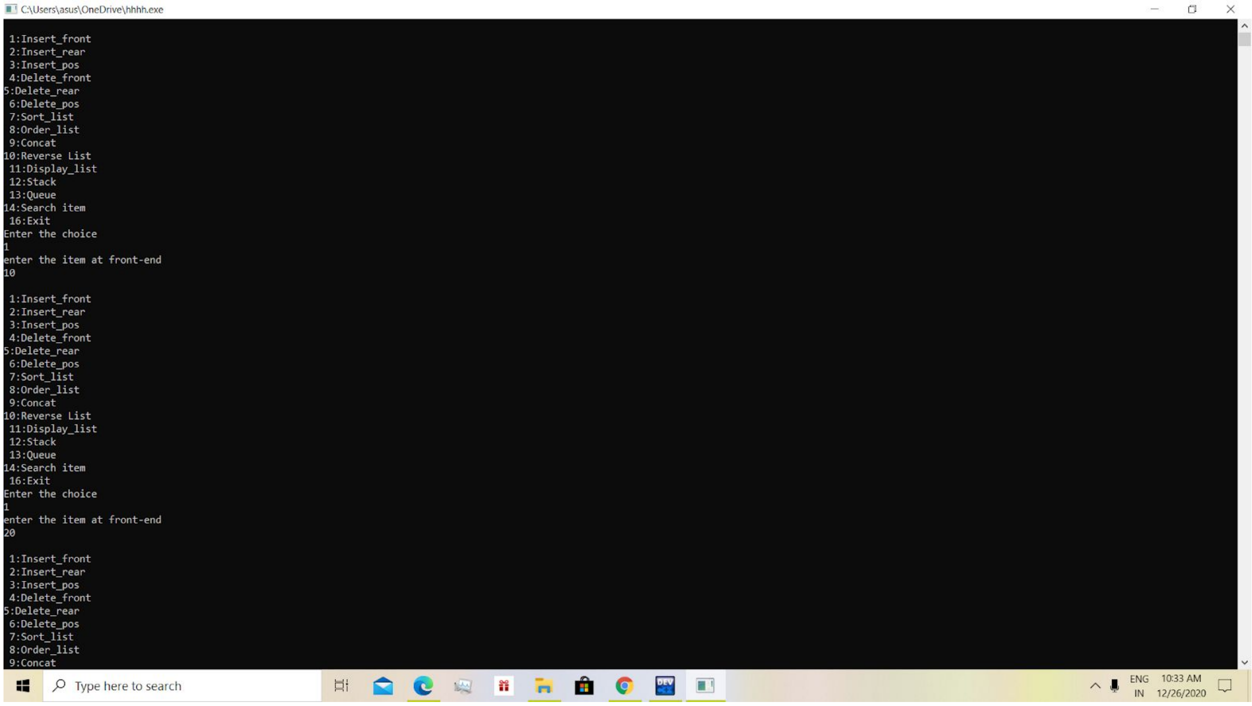
d) Deletion of first element, specified element and last element in the list.
e) Sort the linked list.
f) Reverse the linked list.
g) Concatenation of two linked lists
h) Stack and Queue Implementation

```
#include<stdio.h>
#include<conio.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)
{
```

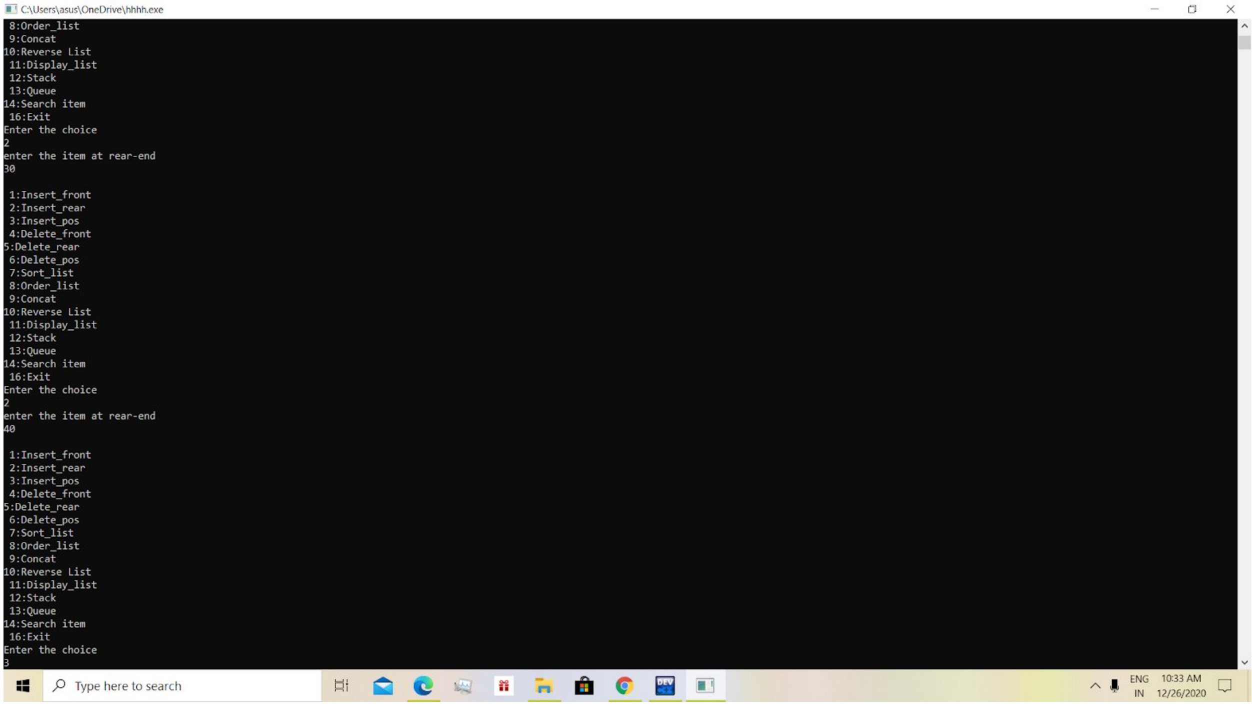
```
NODE temp;
if(first==NULL)
{
printf("List is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("Item deleted at front-end is=%d\n",first->info);
free(first);
return temp;
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
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("Item 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 sort(NODE first)
{
int swapped;
NODE ptr1;
NODE lptr = NULL;
if (first == NULL)
return NULL;
do
    {
        swapped = 0;
        ptr1 = first;

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

                int tem = ptr1->info;
                ptr1->info = ptr1->link->info;
                ptr1->link->info = tem;
                    swapped = 1;
            }
            ptr1 = ptr1->link;
        }
        lptr = ptr1;
    } while (swapped);
}

void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("List empty cannot display items\n");
 else
 printf("Contents of the list:\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;
}

void main()
{
int item,choice,key,n,i,m;
NODE first=NULL,a,b,third,forth;
for(;;)
{
printf("\n 1:Insert_front\n 2:Insert_rear\n 3:Insert_pos\n 4:Delete_front\n" );
printf("5:Delete_rear\n 6:Delete_pos\n 7:Sort_list\n 8:Order_list\n 9:Concat\n");
printf("5:Order_list\n6:Sort_list\n7:Display_list\n8:Concat\n9:Reverse\n10:Stack\n11:Queue\n12:Exit
\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 the item to be inserted in ordered_list\n");
             scanf("%d",&item);
             first=order_list(item,first);
             break;
             case 6:sort(first);
    break;
  case 7:display(first);
```

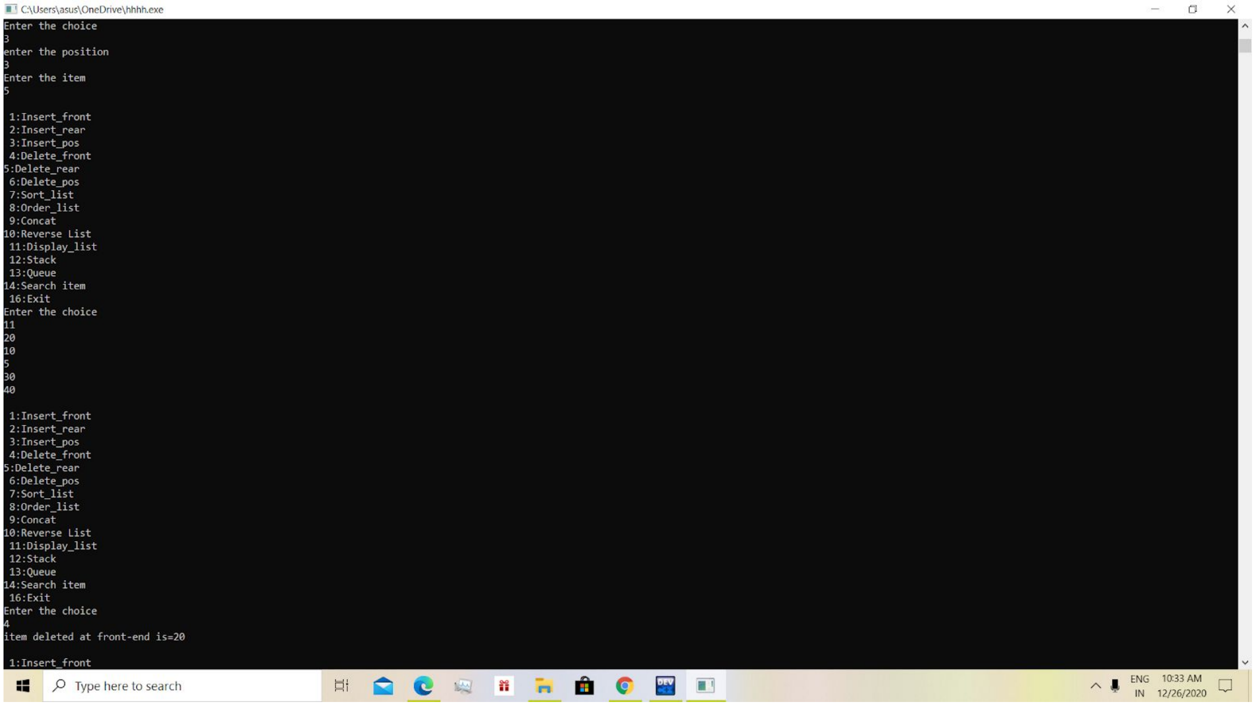
```
        break;
 case 8:printf("Enter the no of nodes in 1\n");
                scanf("%d",&n);
                a=NULL;
                for(i=0;i<n;i++)
                 {
                  printf("Enter the item\n");
                  scanf("%d",&item);
                  a=insert_rear(a,item);
                 }
                 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 9:first=reverse(first);
                display(first);
                break;
  case 10:

  for(;;)
  {

  printf("1:insert at front\n2:delete at front\n3:display\n4:Exit\n");
  printf("enter the choice to stack an element\n");
  scanf("%d",&choice);
  switch(choice)

{

        case 1:
               printf("Enter an item to insert at front to form a stack\n");
               scanf("%d",&item);
                first=insert_front(first, item);
        break;
               case 2:
                 first=delete_front(first);
        break;
                case 3:

                 display(first);
        break;
         case 4:exit(0);
         break;
         default:printf("Invalid choice\n");
         break;
}

  }
  case 11:
        for(;;)
  {

  printf("1:insert at front\n2:delete at rear\n3:display\n4:Exit\n");
  printf("enter the choice to Queue an element\n");
  scanf("%d",&m);
  switch(m)

{


        case 1:
                printf("Enter an item to insert at front end at queue\n");
                scanf("%d",&item);
                first=insert_front(first, item);
        break;
                case 2:

                 first=delete_rear(first);
        break;
                case 3:
        case 4:exit(0);
         break;
                 display(first);
        break;

         default:printf("Invalid choice\n");
         break;
}

  }
  case 12:exit(0);
            break;
            default:printf("Invalid choice\n");
 }
}

}
```



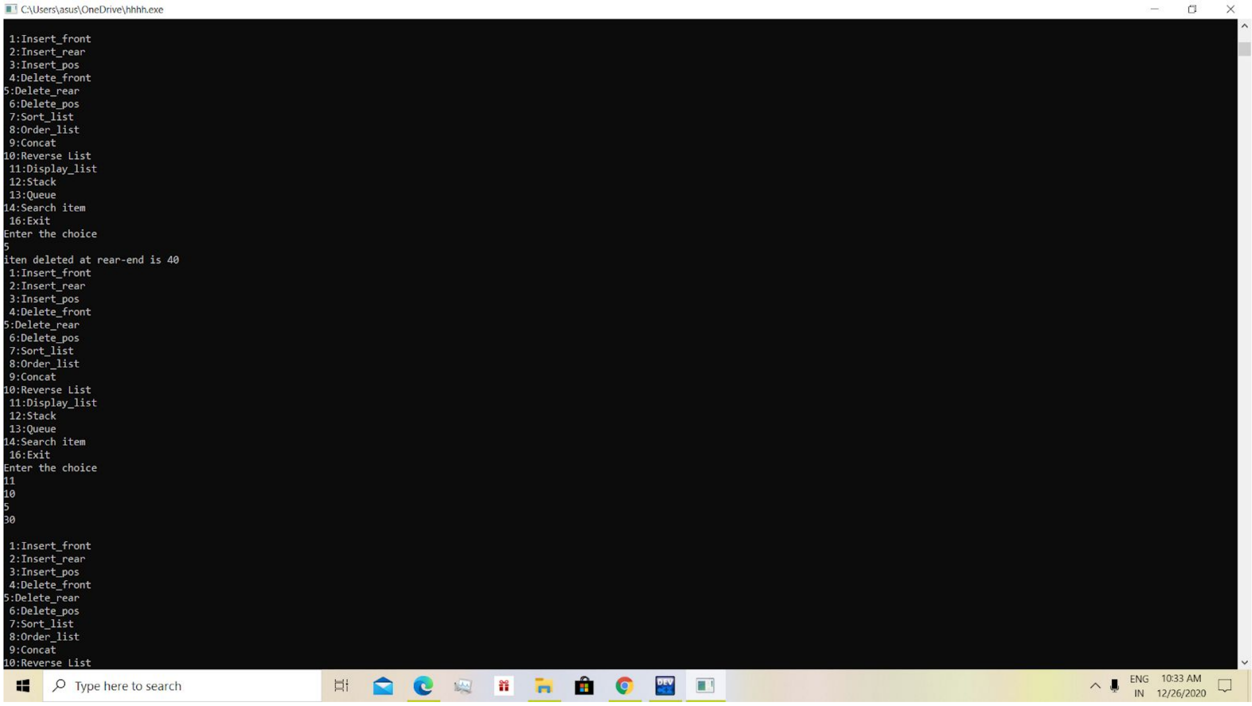
```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
1
enter the item at front-end
10

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
1
enter the item at front-end
20

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
```



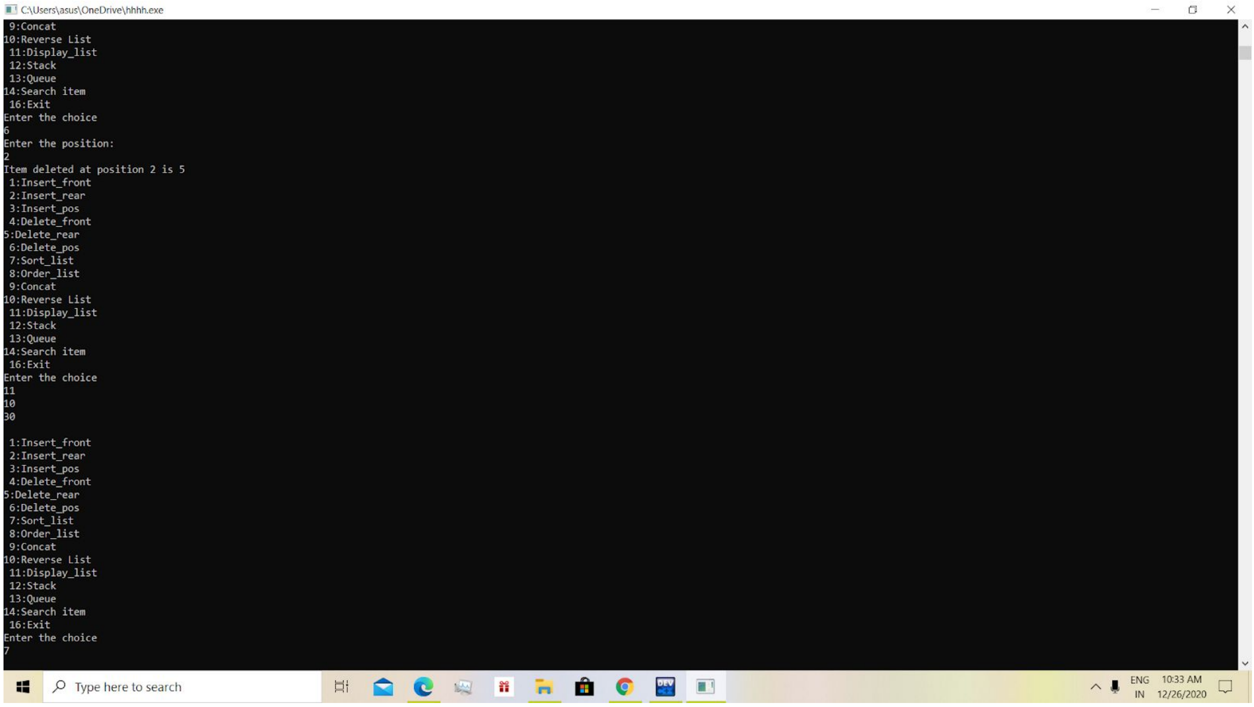
```
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
2
enter the item at rear-end
30

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
2
enter the item at rear-end
40

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
3
```



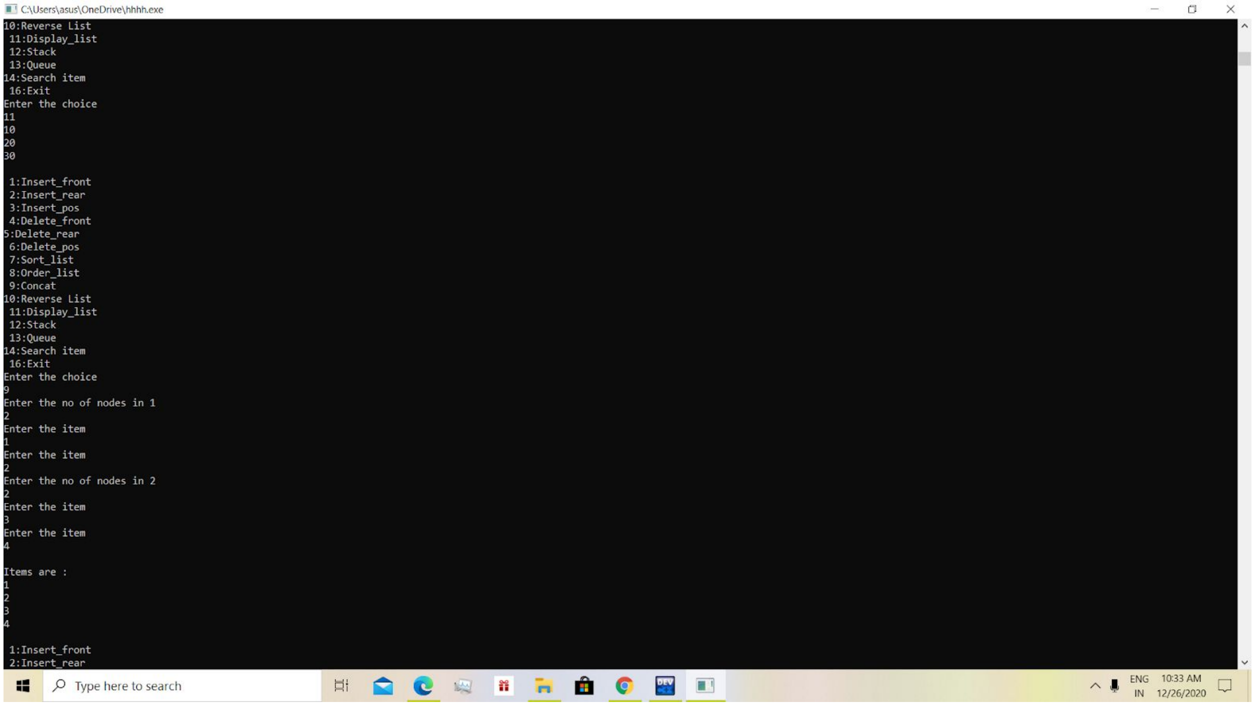
```
Enter the choice
3
enter the position
3
Enter the item
5

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
11
20
10
5
30
40

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
4
item deleted at front-end is=20

 1:Insert_front
```



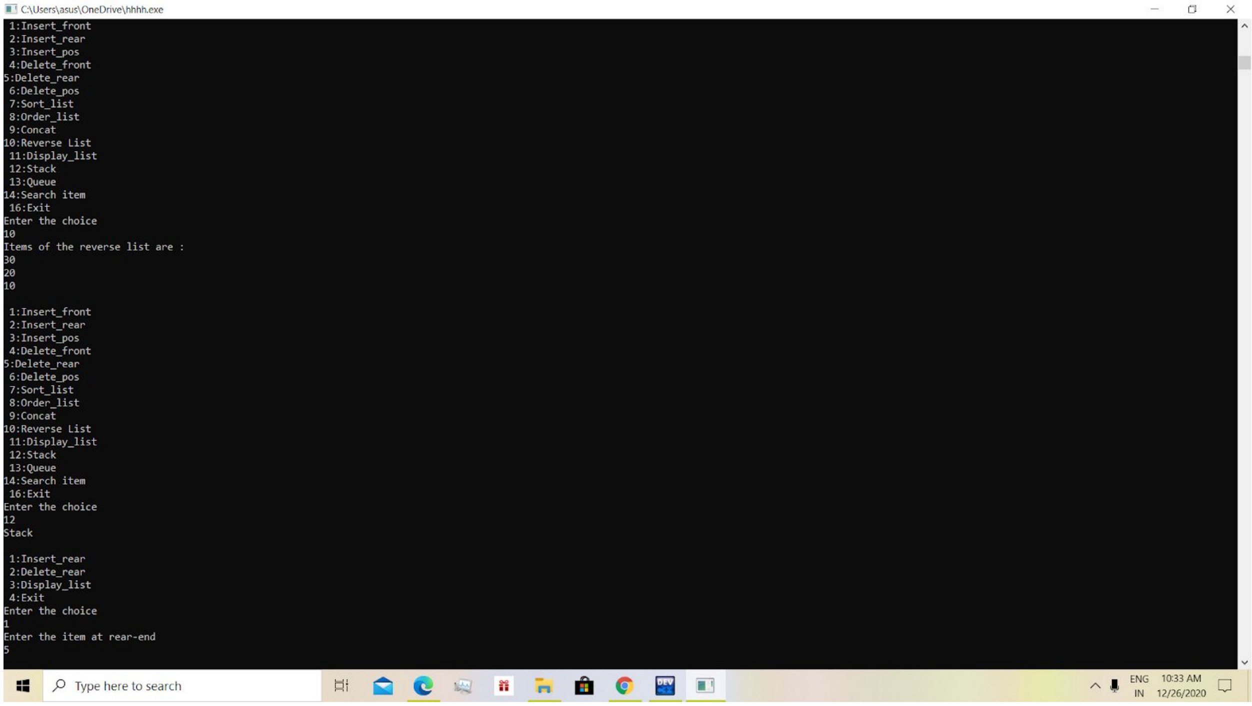
```

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
5
iten deleted at rear-end is 40
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
11
10
5
30

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
```



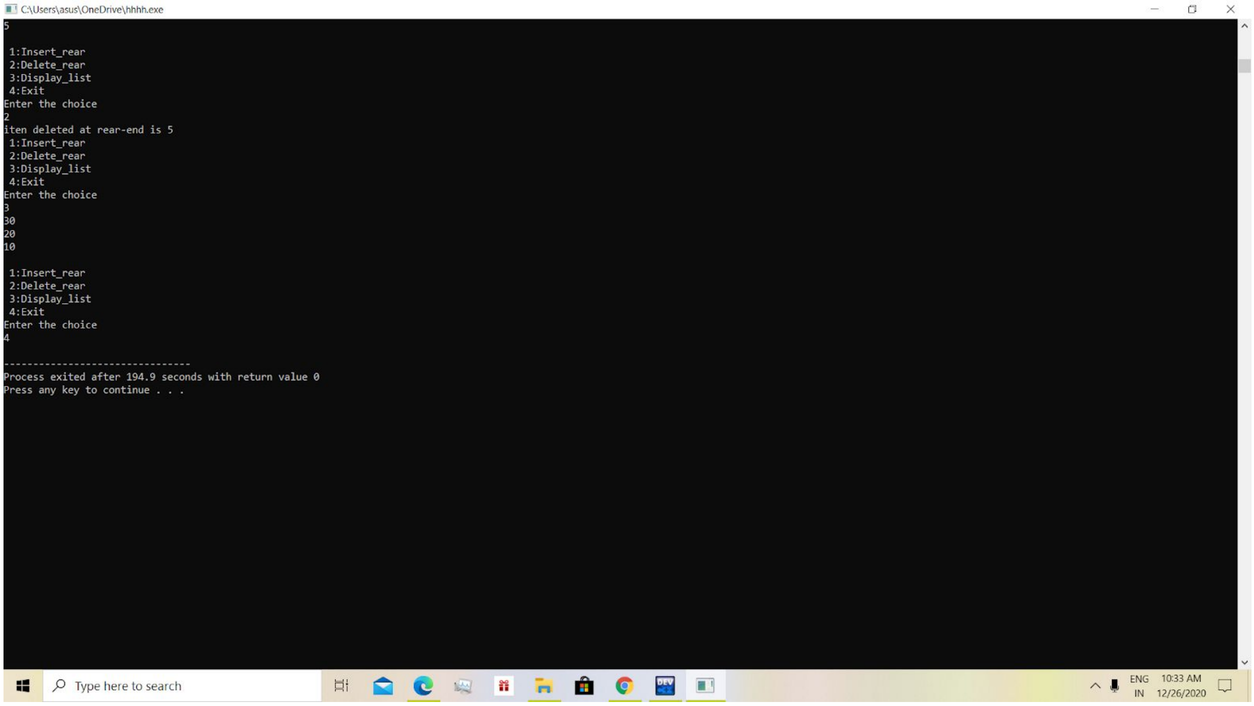
```
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
6
Enter the position:
2
Item deleted at position 2 is 5
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
11
10
30

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
7
```



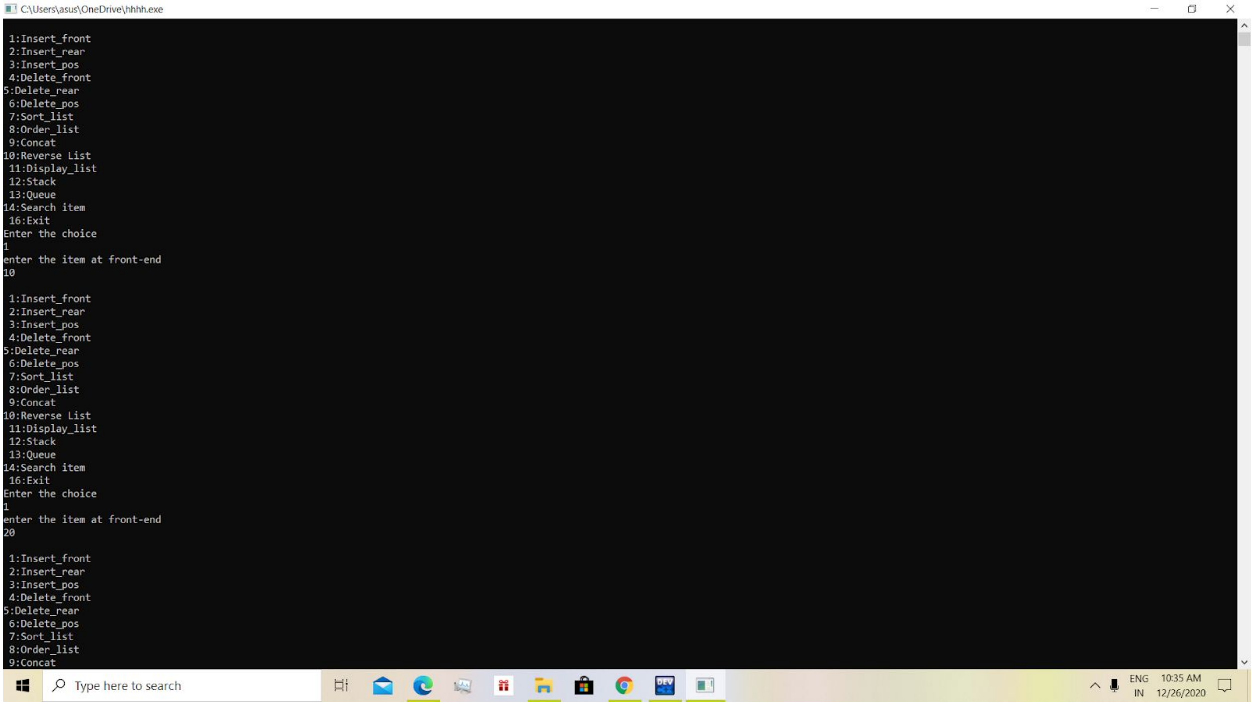
```
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
11
10
20
30

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
9
Enter the no of nodes in 1
2
Enter the item
1
Enter the item
2
Enter the no of nodes in 2
2
Enter the item
3
Enter the item
4

Items are :
1
2
3
4

 1:Insert_front
 2:Insert_rear
```



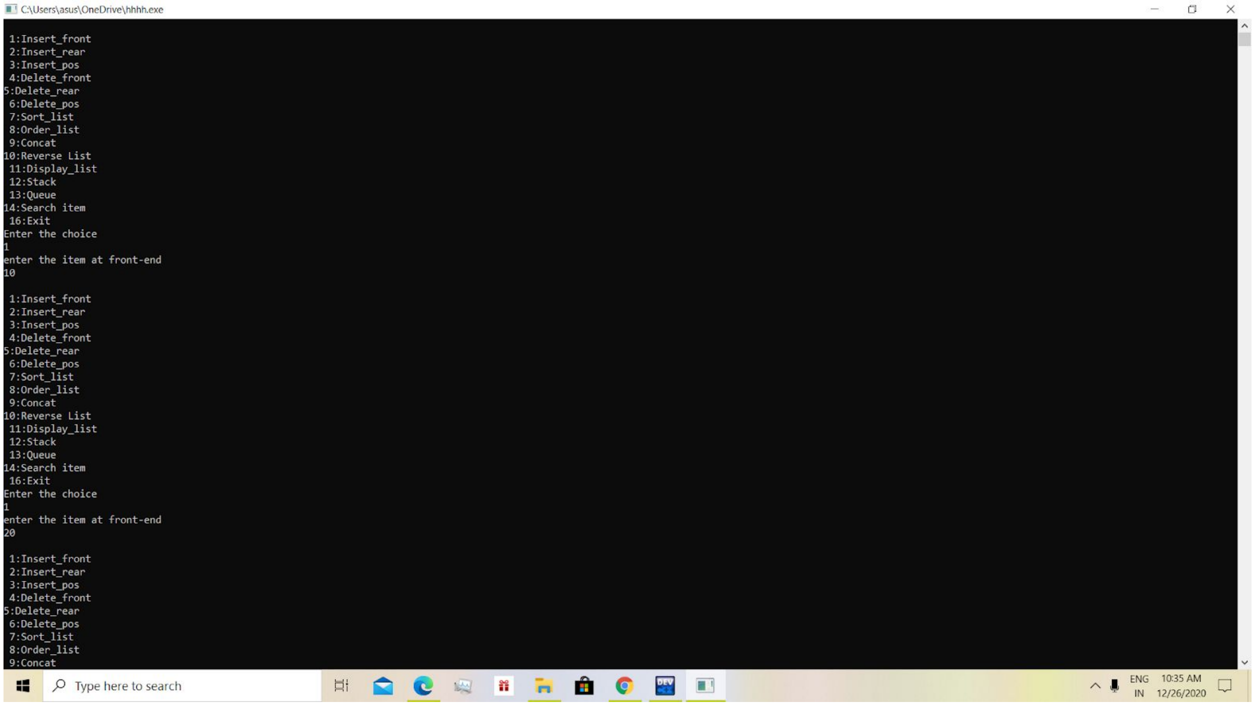
```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
10
Items of the reverse list are :
30
20
10

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
12
Stack

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
5
```



```
5

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
2
iten deleted at rear-end is 5
 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
3
30
20
10

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
4

--------------------------------
Process exited after 194.9 seconds with return value 0
Press any key to continue . . .
```

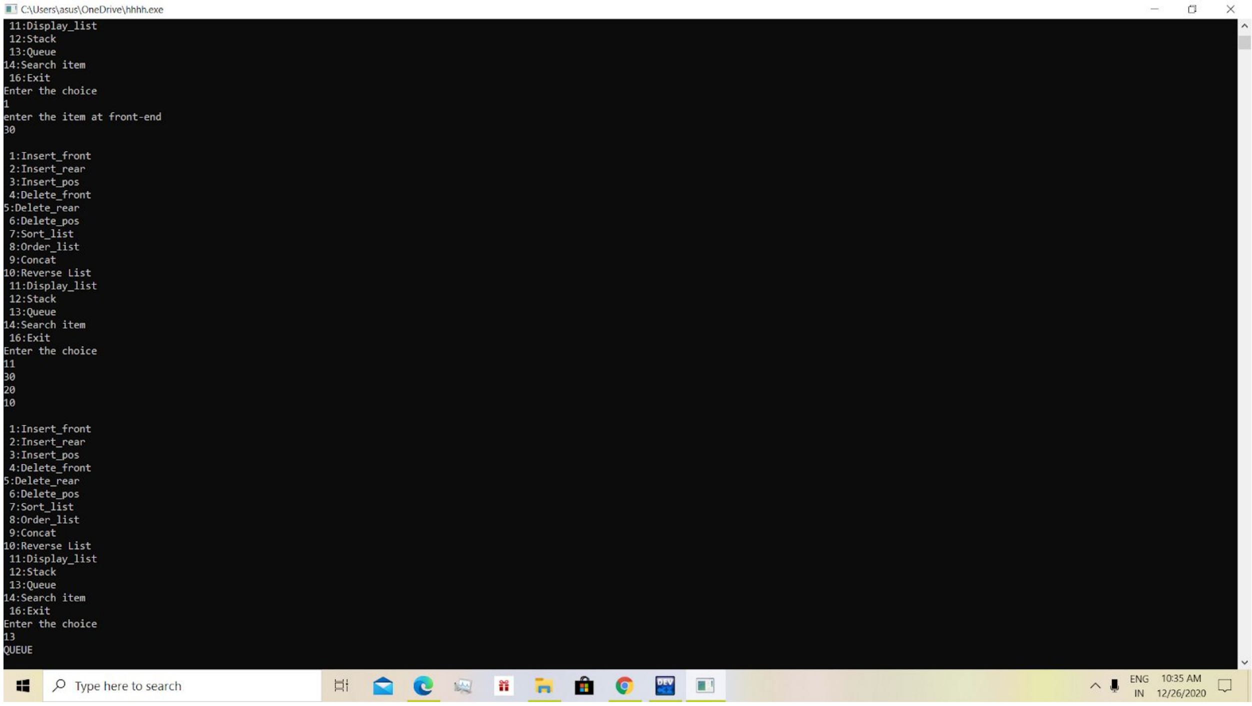


```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
1
enter the item at front-end
10

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
1
enter the item at front-end
20

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
```

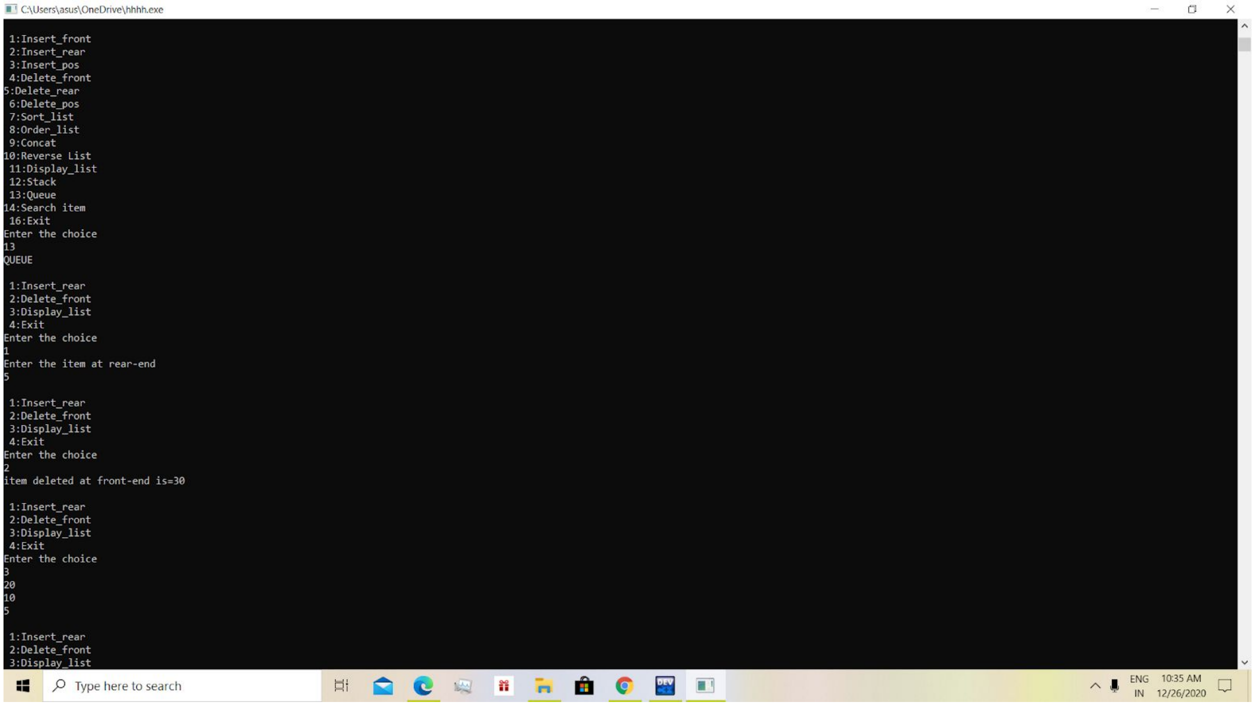

C:\Users\asus\OneDrive\hhhh.exe

```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
1
enter the item at front-end
10

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
1
enter the item at front-end
20

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
```

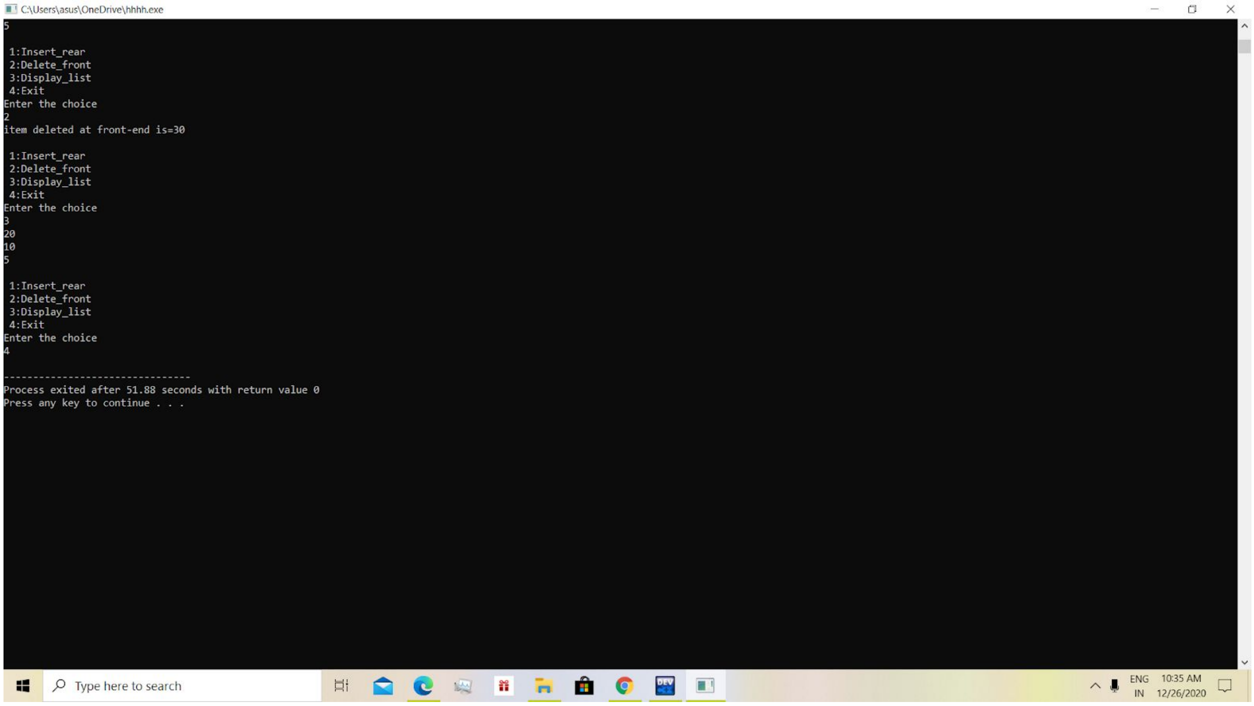
Type here to search
ENG IN 10:35 AM 12/26/2020



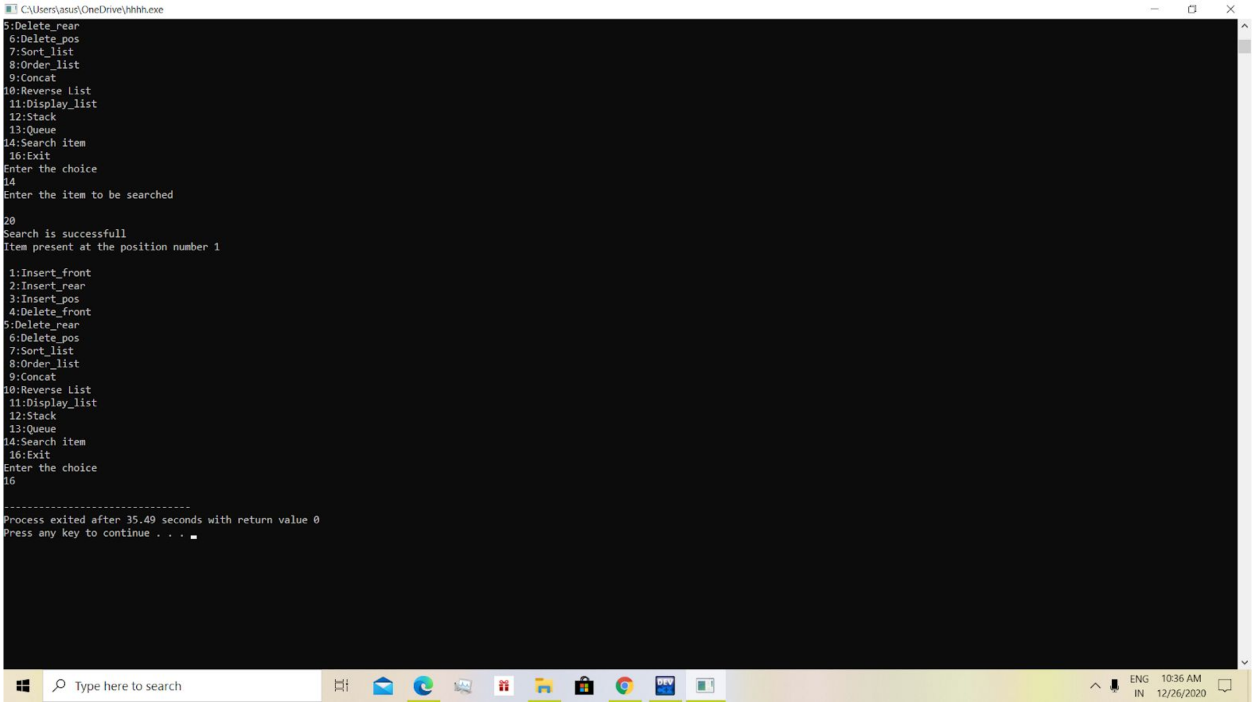
```
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
1
enter the item at front-end
30

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
11
30
20
10

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
13
QUEUE
```



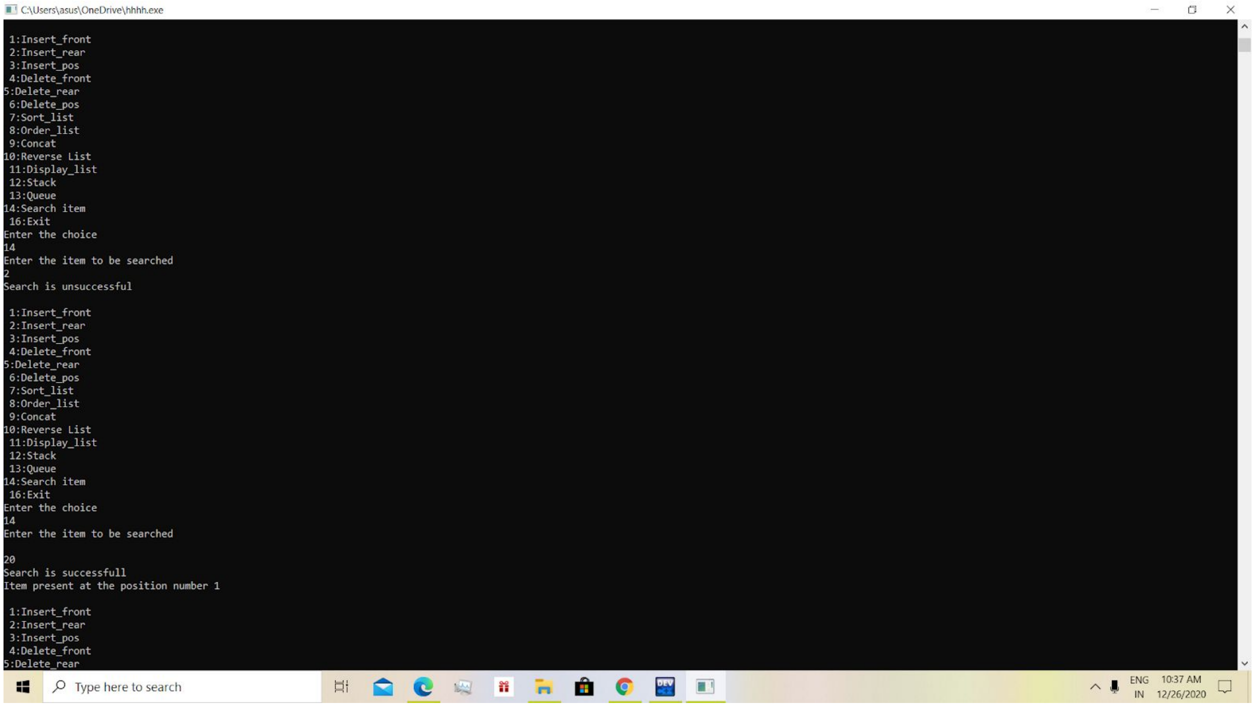
```

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
13
QUEUE

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

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
2
item deleted at front-end is=30

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
3
20
10
5

 1:Insert_rear
 2:Delete_front
 3:Display_list
```



```
5

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
2
item deleted at front-end is=30

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
3
20
10
5

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
4

--------------------------------
Process exited after 51.88 seconds with return value 0
Press any key to continue . . .
```



```
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
14
Enter the item to be searched

20
Search is successfull
Item present at the position number 1

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
16

--------------------------------
Process exited after 35.49 seconds with return value 0
Press any key to continue . . .
```



```
 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
14
Enter the item to be searched
2
Search is unsuccessful

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
 6:Delete_pos
 7:Sort_list
 8:Order_list
 9:Concat
10:Reverse List
 11:Display_list
 12:Stack
 13:Queue
14:Search item
 16:Exit
Enter the choice
14
Enter the item to be searched

20
Search is successfull
Item present at the position number 1

 1:Insert_front
 2:Insert_rear
 3:Insert_pos
 4:Delete_front
5:Delete_rear
```

6)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

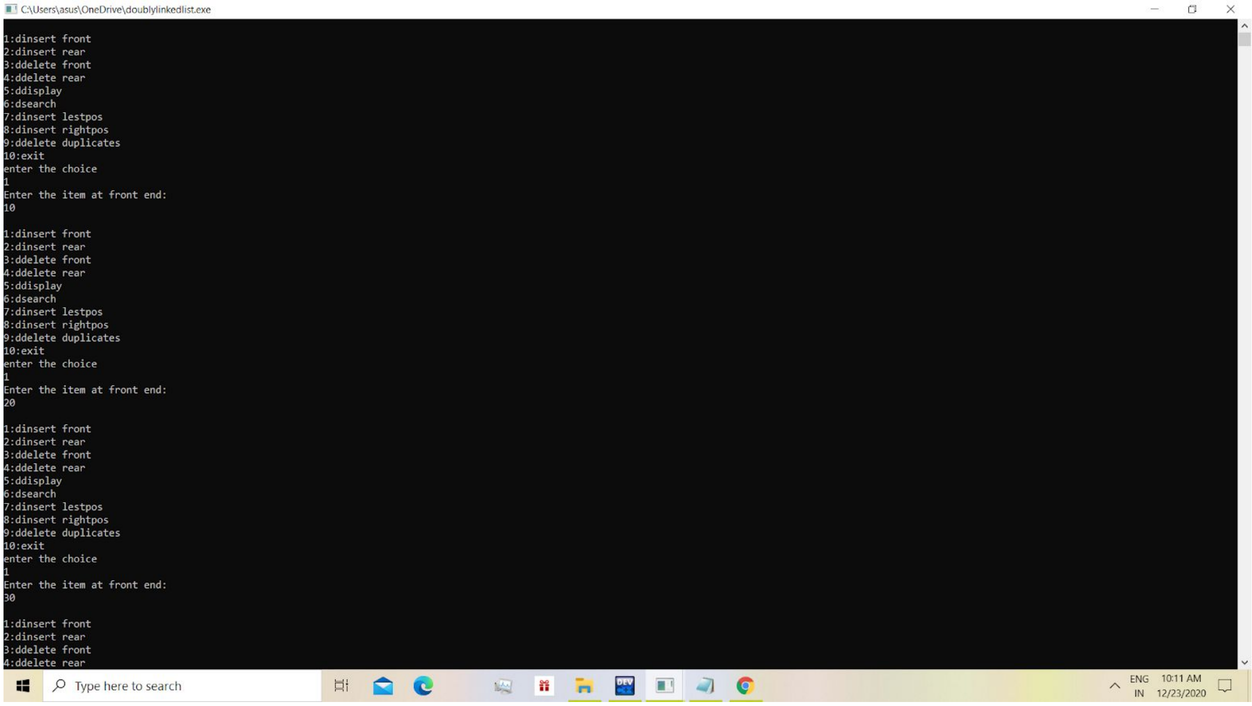
```
#include <stdio.h>
#include <stdlib.h>
struct node
{
	int info;
	struct node *rlink;
	struct node *llink;
};
typedef struct node *NODE;
NODE getnode()
{
	NODE x;
	x=(NODE)malloc(sizeof(struct node));
	if (x==NULL)
	{
		printf("Memory full\n");
		exit(0);
	}
	return x;
}
```

```
NODE dinsert_front(int item,NODE head)
{
        NODE temp,cur;
        temp=getnode();
        temp->info=item;
        temp->llink=NULL;
        temp->rlink=NULL;
        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;
        temp->llink=NULL;
        temp->rlink=NULL;
        cur=head->llink;
        head->llink=temp;
        temp->rlink=head;
        cur->rlink=temp;
        temp->llink=cur;
        return head;
}
NODE ddelete_front(NODE head)
{
        NODE cur,next;
        if (head->rlink==head)
        {
                printf("List is empty\n");
                return head;
        }
        cur=head->rlink;
        next=cur->rlink;
        head->rlink=next;
        next->llink=head;
        printf("Item deleted at the front end is:%d\n",cur->info);
        free(cur);
        return head;
}
```

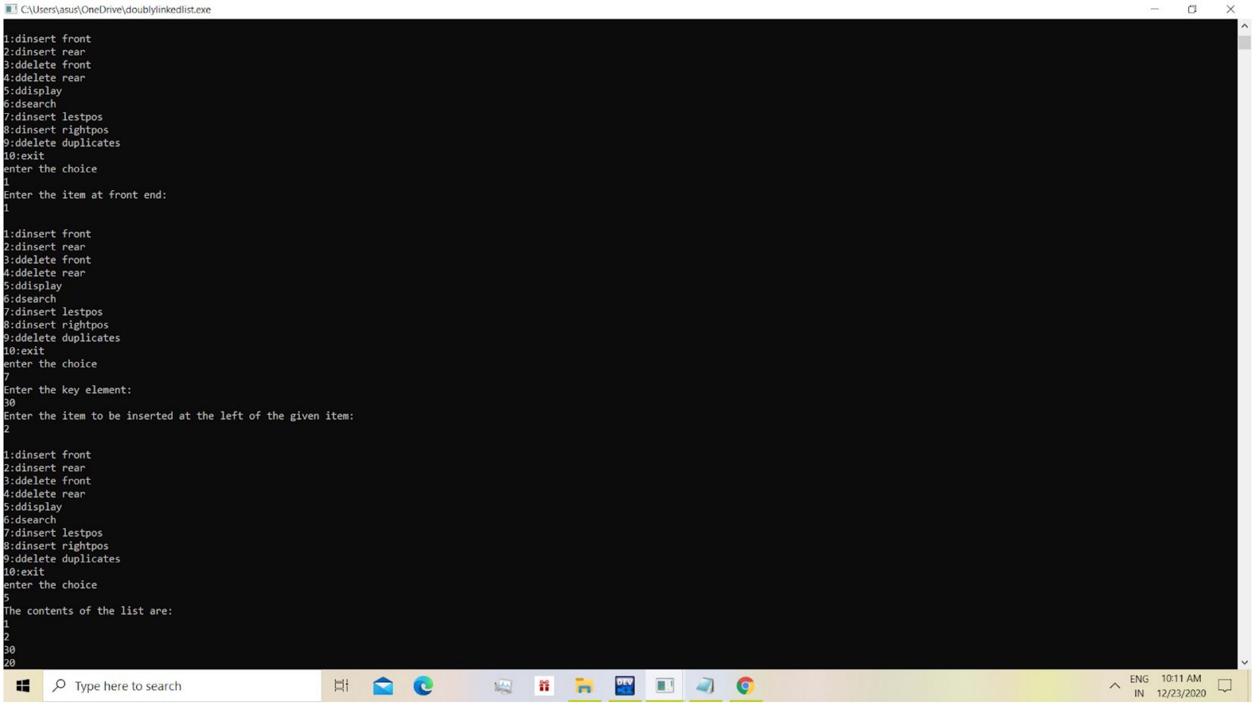
```
NODE ddelete_rear(NODE head)
{
        NODE cur,prev;
        if (head->rlink==head)
        {
                printf("List is empty\n");
                return head;
        }
        cur=head->llink;
        prev=cur->llink;
        prev->rlink=head;
        head->llink=prev;
        printf("Item deleted at the rear end is:%d\n",cur->info);
        free(cur);
        return head;
}
void ddisplay(NODE head)
{
        NODE temp;
        if (head->rlink==head)
        {
                printf("List is empty\n");
        }
        printf("The contents of the list are:\n");
        temp=head->rlink;
        while (temp!=head)
        {
                printf("%d\n",temp->info);
                temp=temp->rlink;
        }
}
void dsearch(int key,NODE head)
{
        NODE cur;
        int count;
        if (head->rlink==head)
        {
                printf("List is empty\n");
        }
        cur=head->rlink;
        count=1;
        while (cur!=head && cur->info!=key)
        {
                cur=cur->rlink;
        count++;
    }
    if (cur==head)
    {
        printf("Search unsuccessfull\n");
    }
    else
    {
        printf("Key element found at the position %d\n",count);
    }
}
NODE dinsert_leftpos(int item,NODE head)
{
    NODE cur,prev,temp;
    if (head->rlink==head)
    {
        printf("List is empty\n");
        return head;
    }
    cur=head->rlink;
    while (cur!=head)
    {
        if (cur->info==item)
        {
            break;
        }
        cur=cur->rlink;
    }
    if (cur==head)
    {
        printf("No such item found in the list\n");
        return head;
    }
    prev=cur->llink;
    temp=getnode();
    temp->llink=NULL;
    temp->rlink=NULL;
    printf("Enter the item to be inserted at the left of the given item:\n");
    scanf("%d",&temp->info);
    prev->rlink=temp;
    temp->llink=prev;
    temp->rlink=cur;
    cur->llink=temp;
    return head;
```

```
}
NODE dinsert_rightpos(int item,NODE head)
{
    NODE temp,cur,next;
    if (head->rlink==head)
    {
        printf("List is empty\n");
        return head;
    }
    cur=head->rlink;
    while (cur!=head)
    {
        if (cur->info==item)
        {
            break;
        }
        cur=cur->rlink;
    }
    if (cur==head)
    {
        printf("No such item found in the list\n");
        return head;
    }
    next=cur->rlink;
    temp=getnode();
    temp->llink=NULL;
    temp->rlink=NULL;
    printf("Enter the item to be inserted at the right of the given item:\n");
    scanf("%d",&temp->info);
    cur->rlink=temp;
    temp->llink=cur;
    next->llink=temp;
    temp->rlink=next;
    return head;
}
NODE ddelete_duplicates(int item,NODE head)
{
    NODE prev,cur,next;
    int count=0;
    if (head->rlink==head)
    {
        printf("List is empty\n");
        return head;
    }
    cur=head->rlink;
    while (cur!=head)
    {
        if (cur->info!=item)
        {
            cur=cur->rlink;
        }
        else
        {
            count++;
            if (count==1)
            {
                cur=cur->rlink;
                continue;
            }
            else
            {
                prev=cur->llink;
                next=cur->rlink;
                prev->rlink=next;
                next->llink=prev;
                free(cur);
                cur=next;
            }
        }
    }
    if (count==0)
    {
        printf("No such item found in the list\n");
    }
    else
    {
        printf("Removed all the duplicate elements of the given item successfully\n");
    }
    return head;
}
int main()
{
NODE head;
int item, choice,key;
head=getnode();
head->llink=head;
head->rlink=head;
for(;;)
```

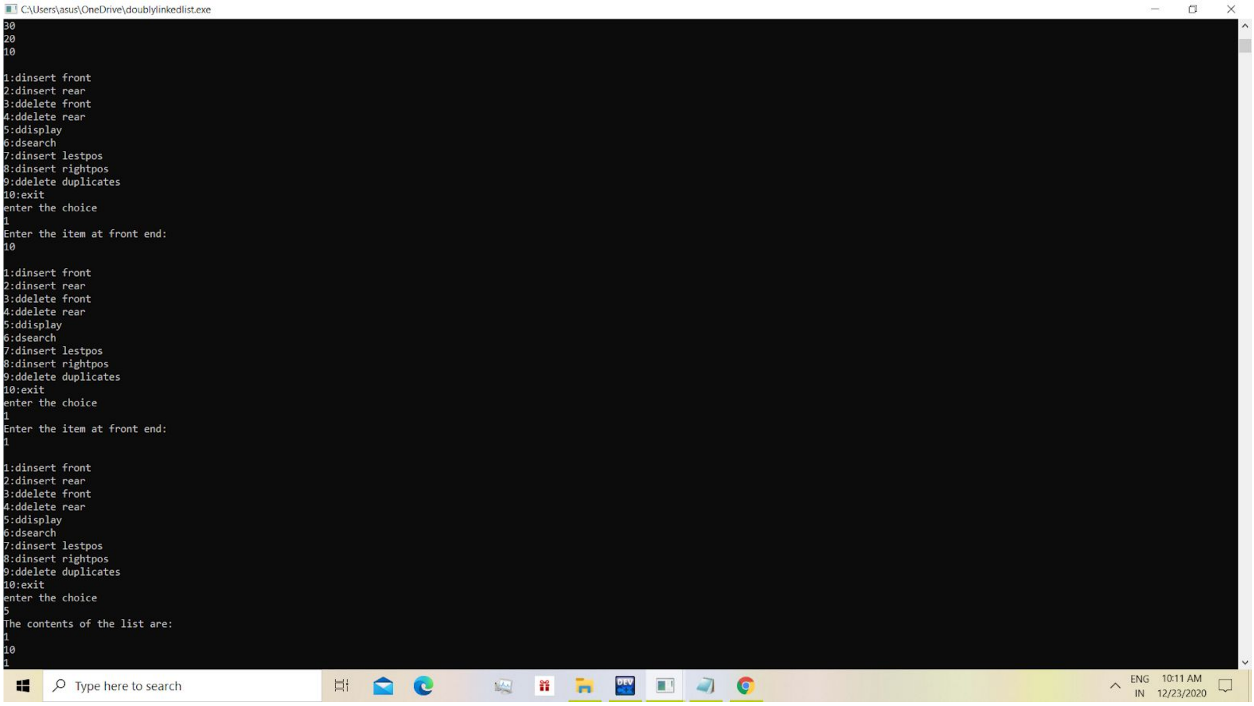
```
{
    printf("\n1:dinsert front\n2:dinsert rear\n3:ddelete front\n4:ddelete
rear\n5:ddisplay\n6:dsearch\n7:dinsert lestpos\n8:dinsert rightpos\n9:ddelete
duplicates\n10:exit\n");
    printf("enter the choice\n");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1: printf("Enter the item at front end:\n");
                scanf("%d",&item);
                head=dinsert_front(item,head);
                break;
        case 2: printf("Enter the item at rear end:\n");
                scanf("%d",&item);
                head=dinsert_rear(item,head);
                break;
        case 3:head=ddelete_front(head);
             break;
        case 4:head=ddelete_rear(head);
             break;
        case 5:ddisplay(head);
             break;
     case 6:printf("Enter the key element to be searched:\n");
             scanf("%d",&key);
             dsearch(key,head);
             break;
     case 7:printf("Enter the key element:\n");
             scanf("%d",&key);
             head=dinsert_leftpos(key,head);
             break;
        case 8:printf("Enter the key element:\n");
             scanf("%d",&key);
             head=dinsert_rightpos(key,head);
             break;
        case 9:printf("Enter the key element whose duplicates should be removed:\n");
             scanf("%d",&key);
             head=ddelete_duplicates(key,head);
             break;
        default:exit(0);
        }
    }
    return 0;
}
```



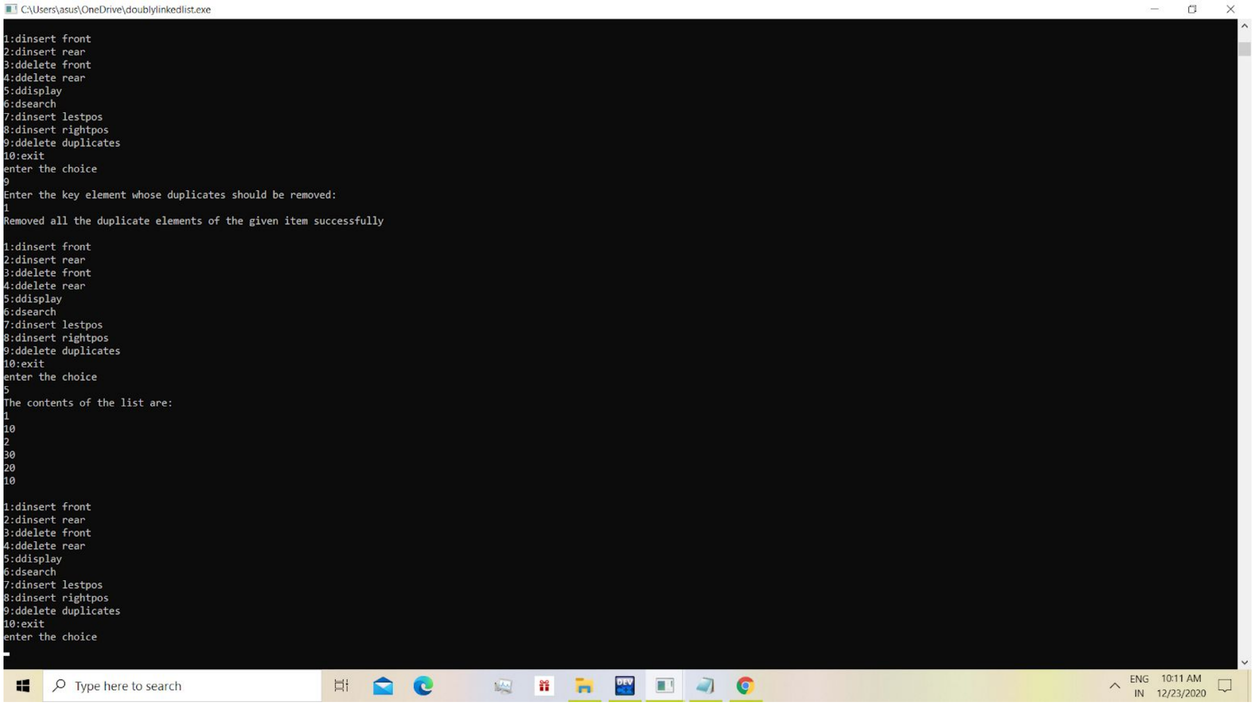
```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:exit
enter the choice
1
Enter the item at front end:
10

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:exit
enter the choice
1
Enter the item at front end:
20

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:exit
enter the choice
1
Enter the item at front end:
30

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
```



```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:exit
enter the choice
1
Enter the item at front end:
1

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:exit
enter the choice
7
Enter the key element:
30
Enter the item to be inserted at the left of the given item:
2

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:exit
enter the choice
5
The contents of the list are:
1
2
30
20
```



```
30
20
10

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:exit
enter the choice
1
Enter the item at front end:
10

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:exit
enter the choice
1
Enter the item at front end:
1

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:exit
enter the choice
5
The contents of the list are:
1
10
1
```



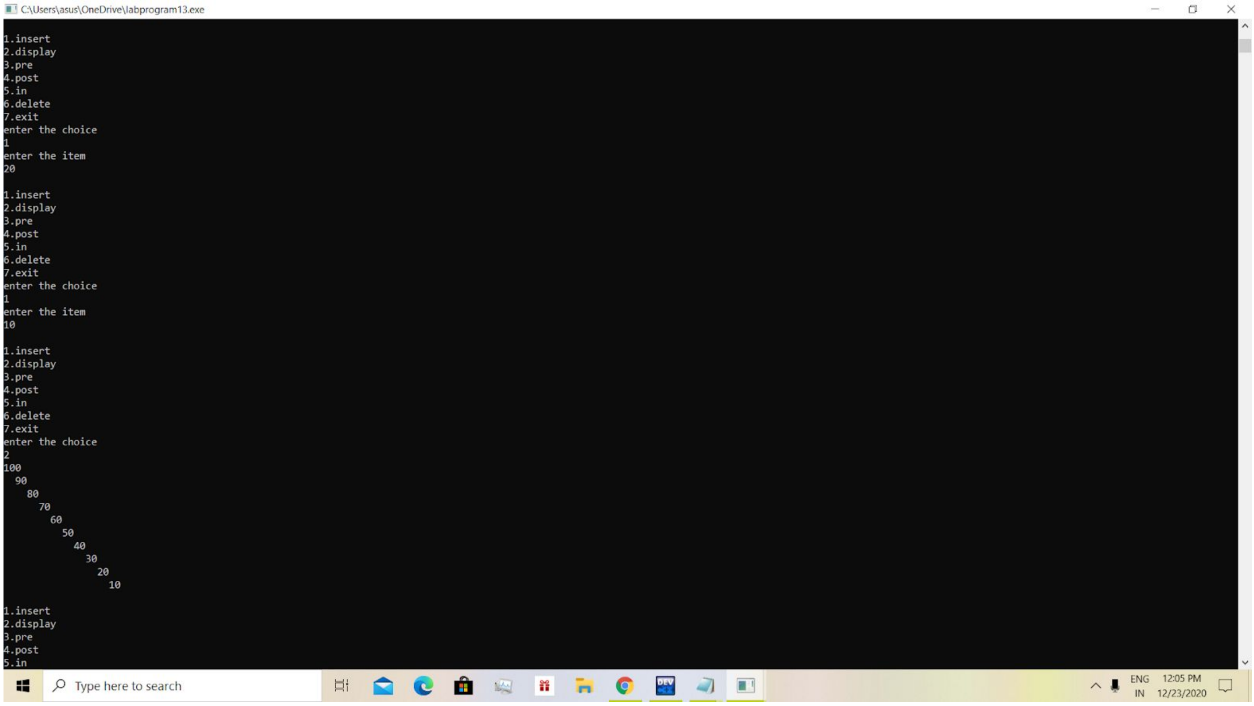
```

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:exit
enter the choice
9
Enter the key element whose duplicates should be removed:
1
Removed all the duplicate elements of the given item successfully

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:exit
enter the choice
5
The contents of the list are:
1
10
2
30
20
10

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:exit
enter the choice
_
```

7) 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

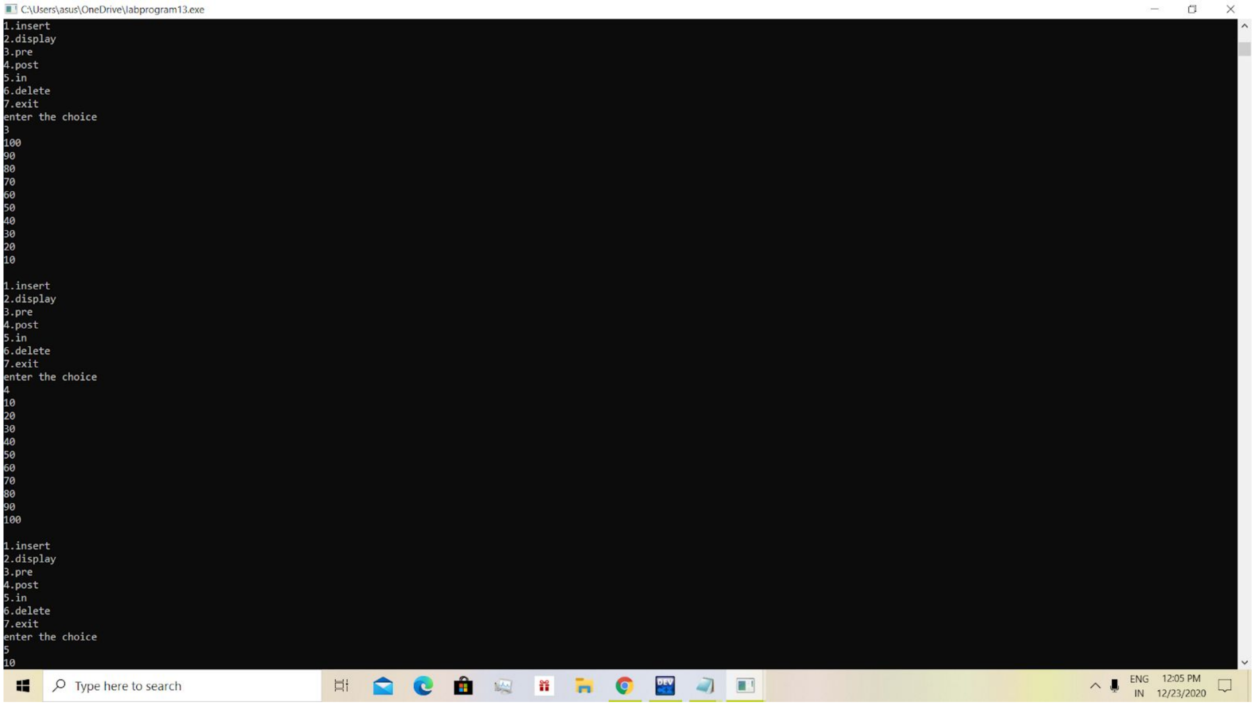
c) To display the elements in the tree

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
#include<process.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);
 }
}
NODE delete(NODE root,int item)
{
NODE cur,parent,q,suc;
if(root==NULL)
{
printf("empty\n");
return root;
}
parent=NULL;
cur=root;
while(cur!=NULL&&item!=cur->info)
{
parent=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(cur==NULL)
{
 printf("not found\n");
 return root;
}
if(cur->llink==NULL)
 q=cur->rlink;
else if(cur->rlink==NULL)
 q=cur->llink;
```

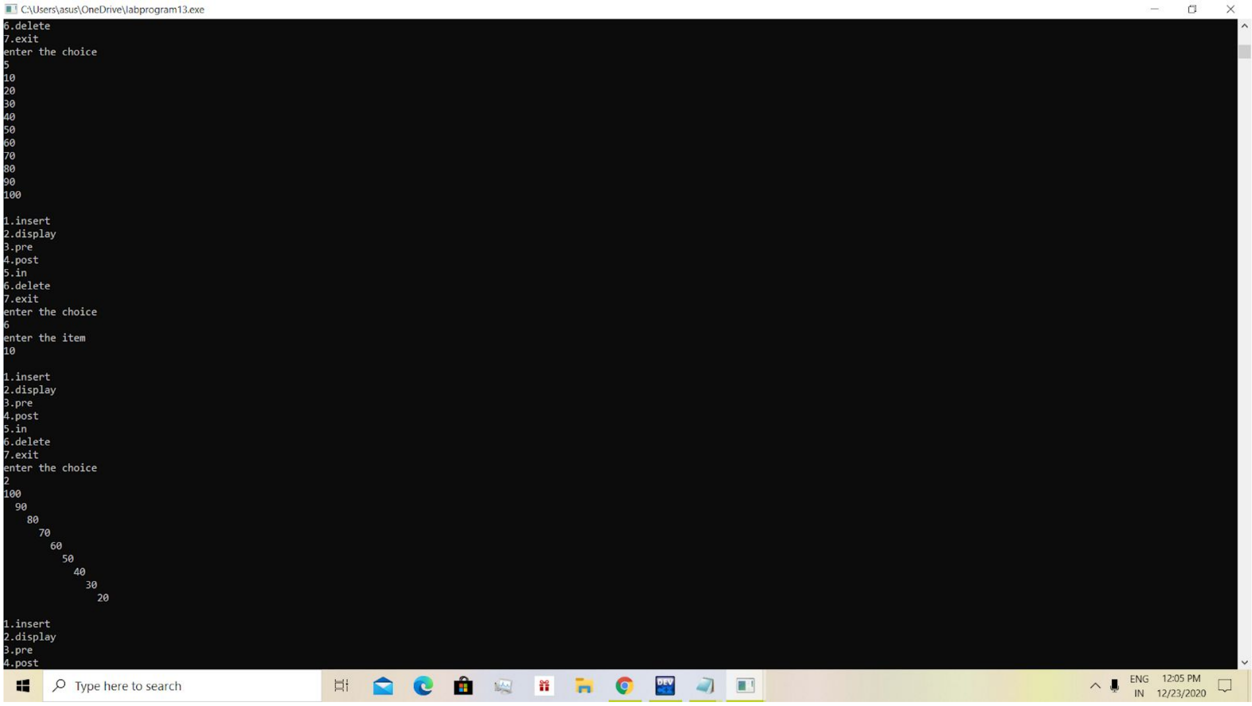
```
else
 {
 suc=cur->rlink;
 while(suc->llink!=NULL)
  suc=suc->llink;
 suc->llink=cur->llink;
 q=cur->rlink;
 }
 if(parent==NULL)
  return q;
 if(cur==parent->llink)
  parent->llink=q;
 else
  parent->rlink=q;
 freenode(cur);
 return root;
 }

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);
  }
 }
void main()
{
int item,choice;
NODE root=NULL;
for(;;)
{
printf("\n1.insert\n2.display\n3.pre\n4.post\n5.in\n6.delete\n7.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:printf("enter the item\n");
                    scanf("%d",&item);
                    root=delete(root,item);
                    break;
  default:exit(0);
                     break;
           }
          }
 }
```



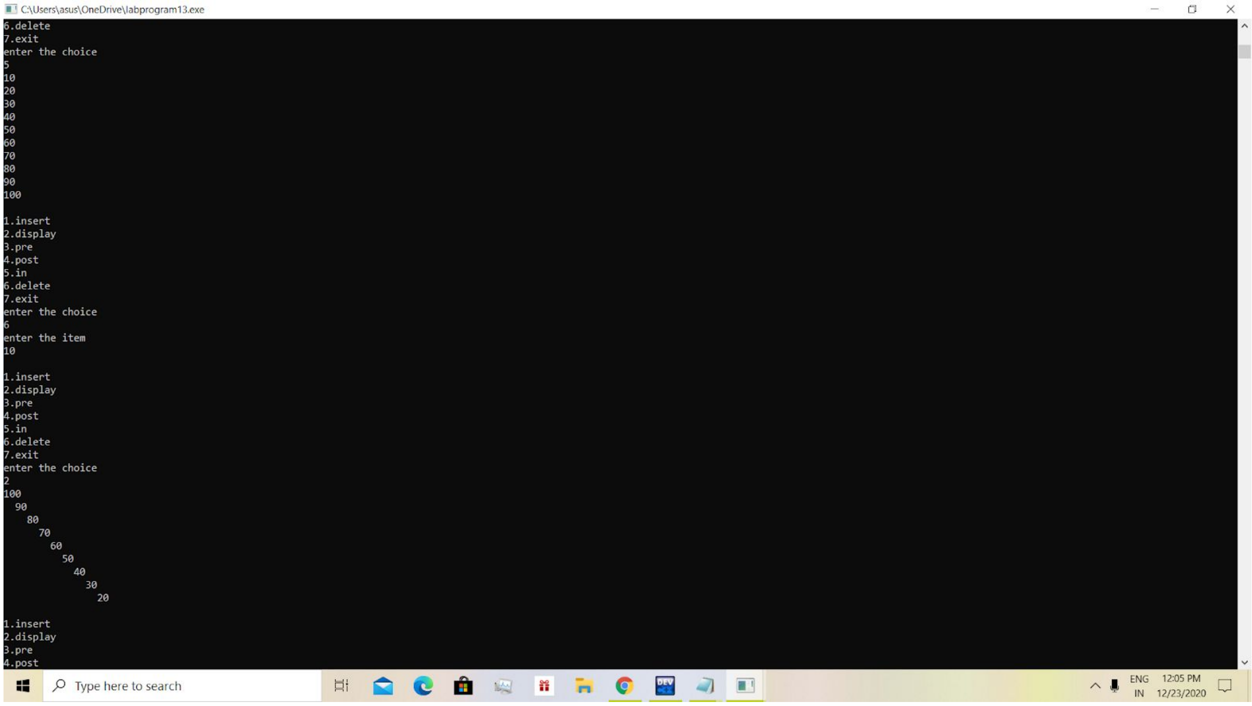
```

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
20

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
100
  90
    80
      70
        60
          50
            40
              30
                20
                  10

1.insert
2.display
3.pre
4.post
5.in
```



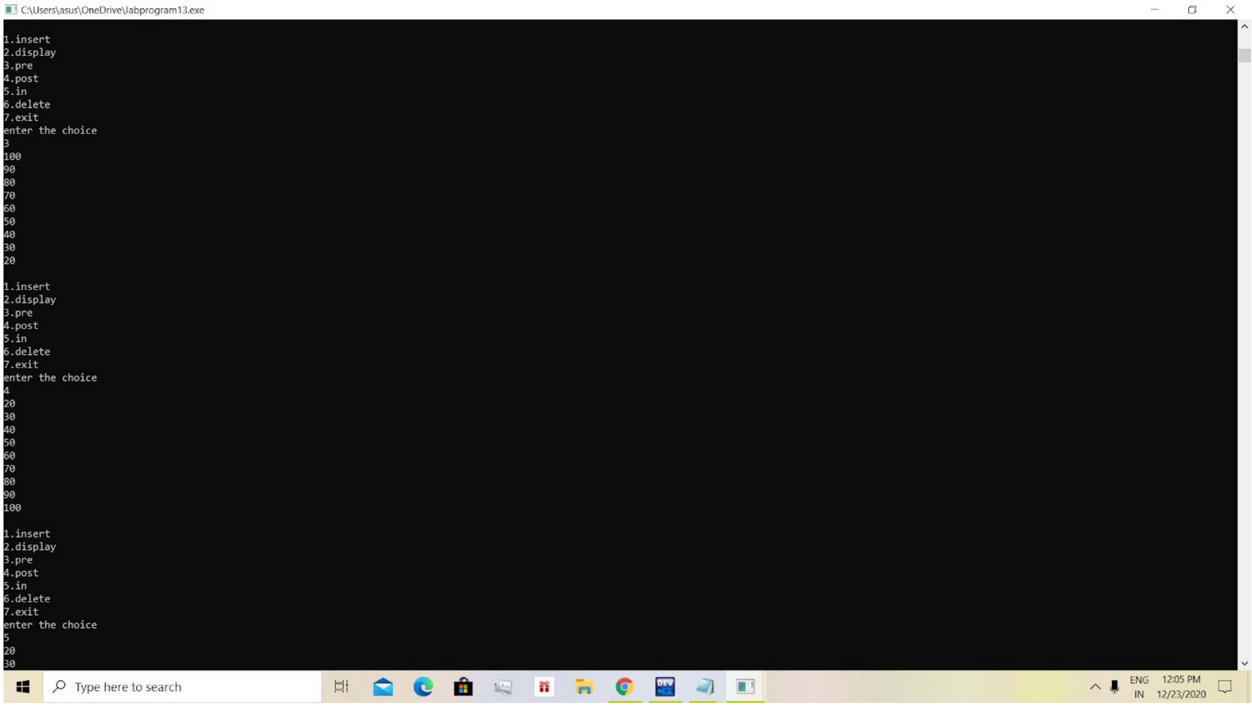
```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
3
100
90
80
70
60
50
40
30
20
10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
10
20
30
40
50
60
70
80
90
100

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
10
```



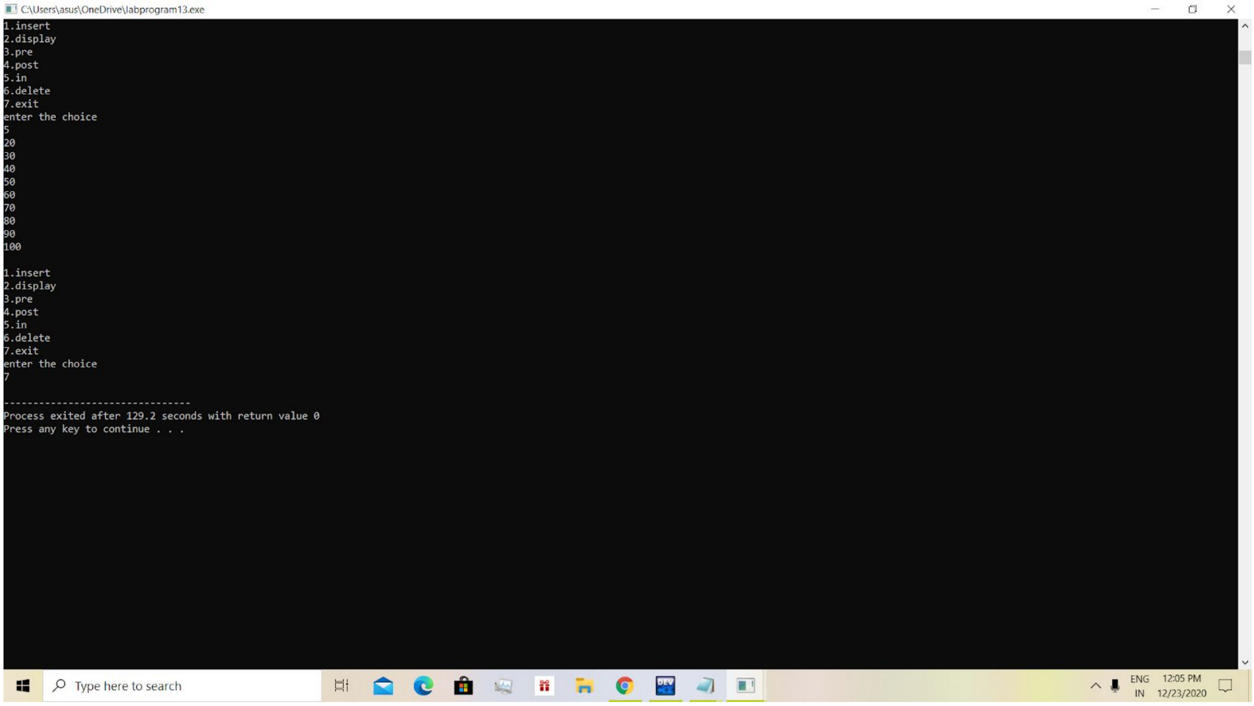
```
6.delete
7.exit
enter the choice
5
10
20
30
40
50
60
70
80
90
100

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
6
enter the item
10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
100
  90
    80
      70
        60
          50
            40
              30
                20

1.insert
2.display
3.pre
4.post
```


C:\Users\asus\OneDrive\labprogram13.exe
Type here to search
ENG
IN
12:05 PM
12/23/2020

```
6.delete
7.exit
enter the choice
5
10
20
30
40
50
60
70
80
90
100

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
6
enter the item
10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
100
  90
    80
      70
        60
          50
            40
              30
                20

1.insert
2.display
3.pre
4.post
```



```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
3
100
90
80
70
60
50
40
30
20

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
20
30
40
50
60
70
80
90
100

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
20
30
```



```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
20
30
40
50
60
70
80
90
100

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
7

--------------------------------
Process exited after 129.2 seconds with return value 0
Press any key to continue . . .
```