DATA STRUCTURES LAB RECORD

Name : Manikanth Lakshman Shetty **USN :** 1BM19CS082

Section : 3-B **Batch :** 2

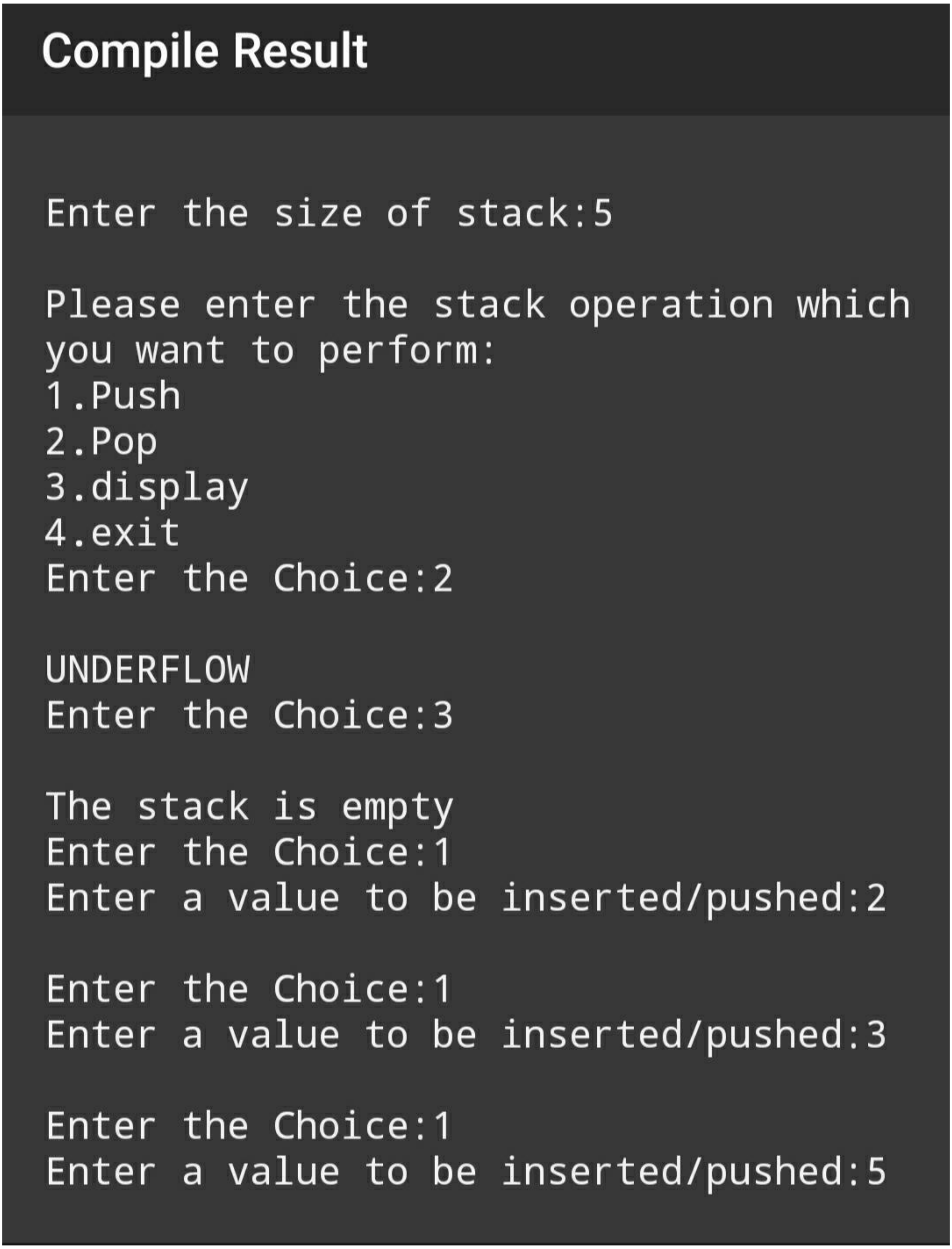
1)**Lab1:**

Write a program to simulate the working of stack using an array with the following :
a) Push b) Pop c) Display
The program should print appropriate messages for stack overflow, stack underflow

```
#include<stdio.h>

#include<stdlib.h>

int stack[50];


int ch;

void push(void);

void pop(void);

void display(void);

int n,top,no,i;

int main()

{

   top=-1;

   printf("\n Enter the size of stack:");

   scanf("%d",&n);

   printf("\n Please enter the stack operation which you want to perform:");
printf("\n 1.Push\n 2.Pop\n 3.display\n 4.exit");
while(ch!='0')
{
   printf("\n Enter the Choice:");
   scanf("%d",&ch);
   switch(ch)
   {
      case 1:
         push();
         break;
      case 2:
         pop();
         break;
      case 3:
         display();
         break;
      case 4:
         exit(0);
         break;
      default:
      {
         printf ("\nINVALID CHOICE!");
      }

   }
}
```

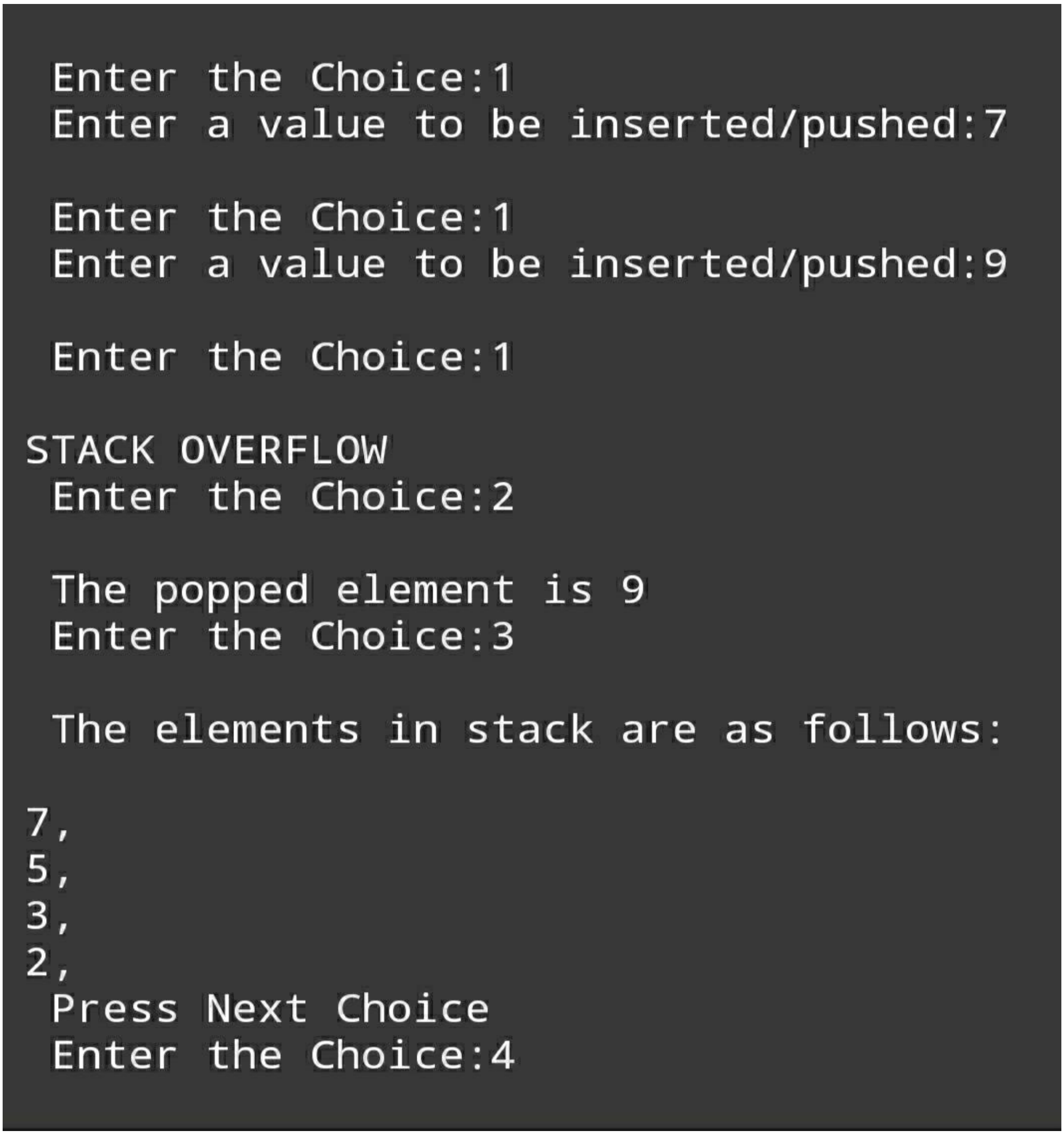
```
    return 0;
}
void push()
{
    if(top>=n-1)
    {
        printf("\nSTACK OVERFLOW");

    }
    else
    {
        printf(" Enter a value to be inserted/pushed:");
        scanf("%d",&no);
        top++;
        stack[top]=no;
    }
}
void pop()
{
    if(top<=-1)
    {
        printf("\n UNDERFLOW");
    }
    else
    {
        printf("\n The popped element is %d",stack[top]);
        top--;
    }
}
void display()
{
    if(top>=0)
    {
        printf("\n The elements in stack are as follows: \n");
        for(i=top;i>=0;i--)
            printf("\n%d\n",stack[i]);
        printf("\n Press Next Choice");
    }
    else
    {
        printf("\n The stack is empty");
    }

}
```

OUTPUT :



Compile Result

```
Enter the size of stack:5

Please enter the stack operation which
you want to perform:
1.Push
2.Pop
3.display
4.exit
Enter the Choice:2

UNDERFLOW
Enter the Choice:3

The stack is empty
Enter the Choice:1
Enter a value to be inserted/pushed:2

Enter the Choice:1
Enter a value to be inserted/pushed:3

Enter the Choice:1
Enter a value to be inserted/pushed:5
```



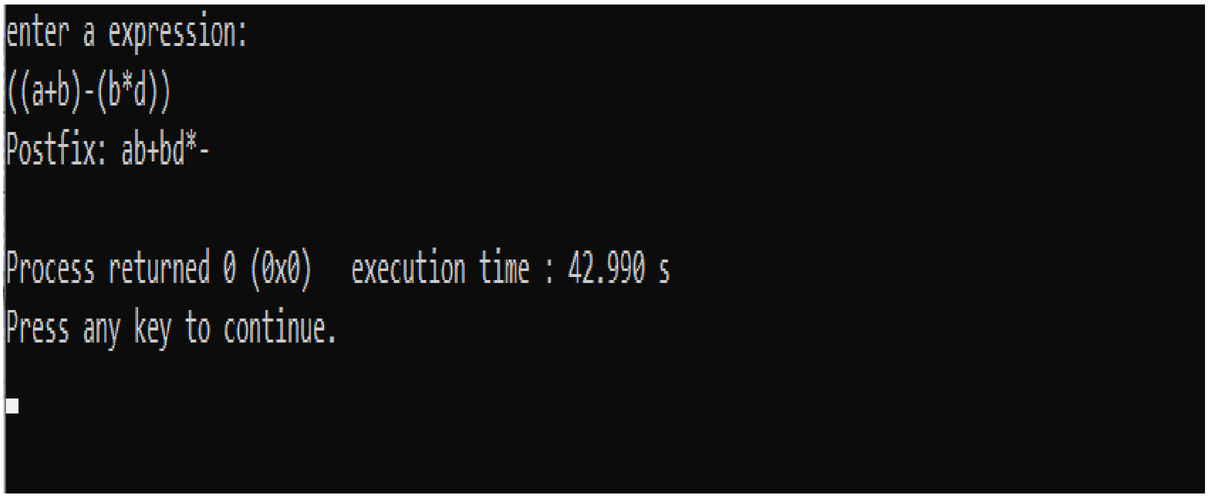
```
 Enter the Choice:1
 Enter a value to be inserted/pushed:7

 Enter the Choice:1
 Enter a value to be inserted/pushed:9

 Enter the Choice:1

STACK OVERFLOW
 Enter the Choice:2

 The popped element is 9
 Enter the Choice:3

 The elements in stack are as follows:

7,
5,
3,
2,
 Press Next Choice
 Enter the Choice:4
```

2)**Lab2 :**

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[]){
    int top,j,i;
    char s[30],postfix[30];
    char 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';
    printf("Postfix: ");
    puts(postfix);
}
int main()
{
    char exp[30];
    printf("enter a expression:\n");
    scanf("%s",exp);
    infix_postfix(exp);
    return 0;
}
```

OUTPUT :



```
enter a expression:
((a+b)-(b*d))
Postfix: ab+bd*-

Process returned 0 (0x0)   execution time : 42.990 s
Press any key to continue.
```

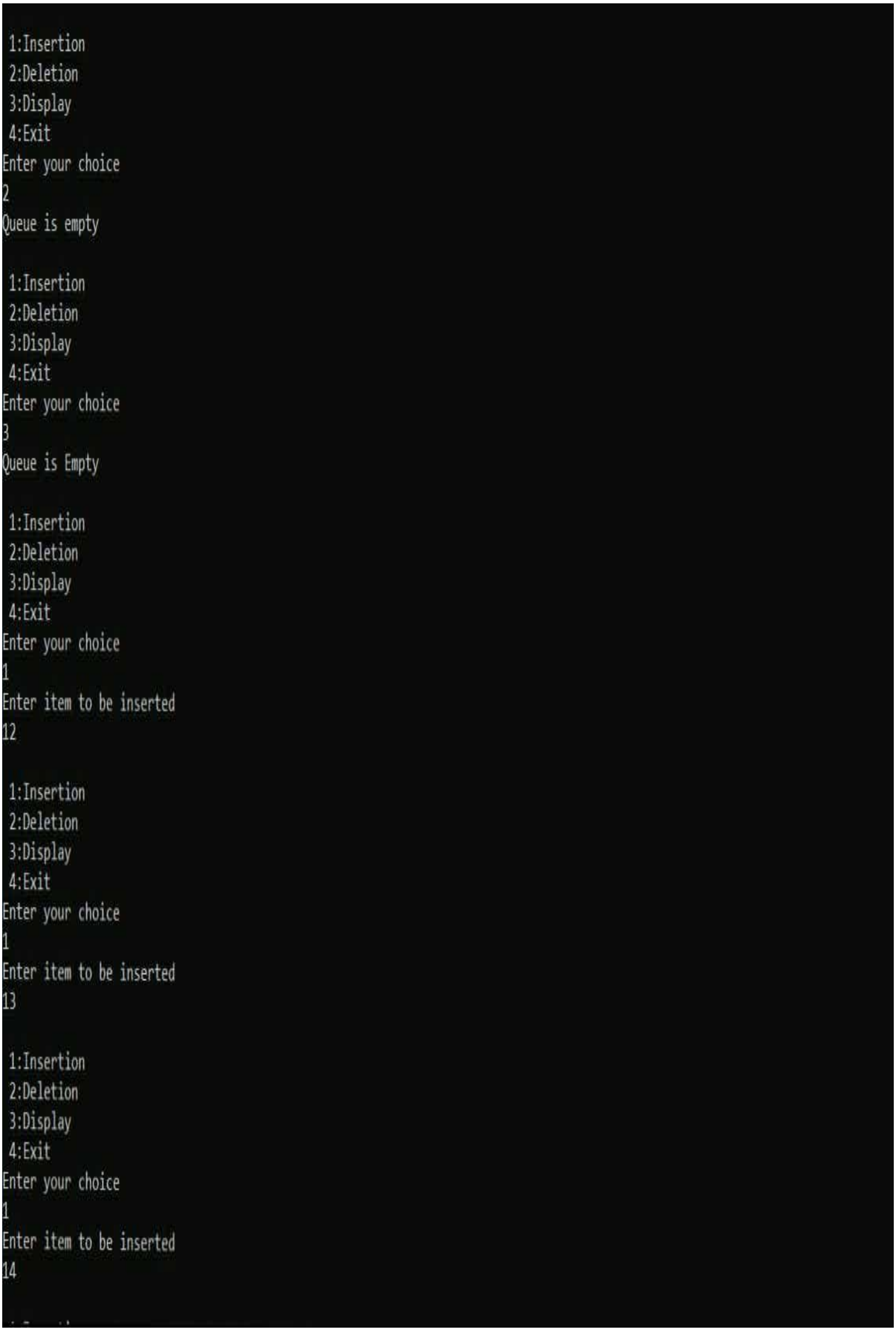
3) **Lab3 :**

WAP to simulate the working of a queue of integers using an array. Provide the following
operations
a) Insert b) Delete c) Display
The program should print appropriate messages for queue empty and queue overflow
Conditions

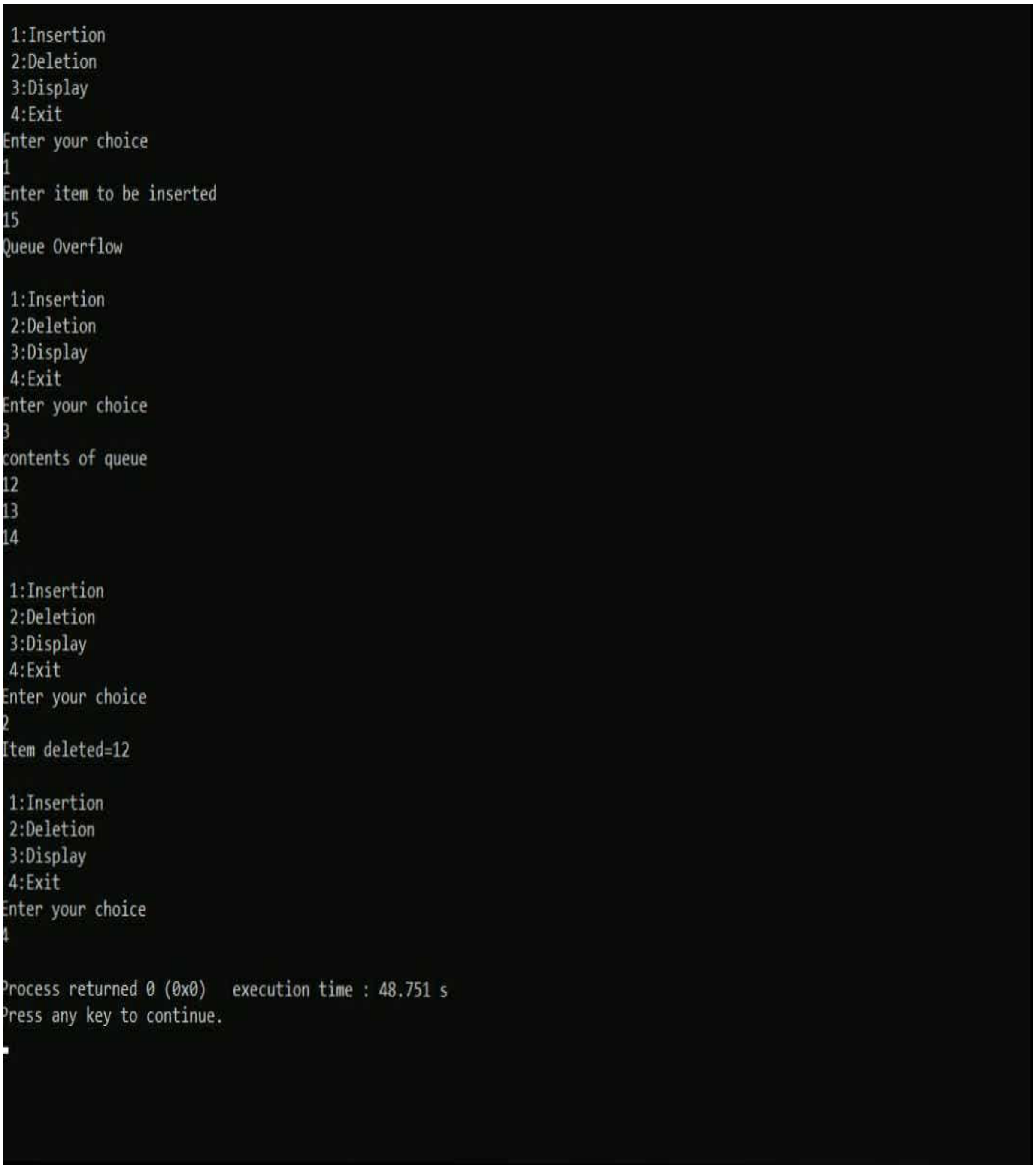
```
#include<stdio.h>
#include<stdlib.h>
#define Que_Size 3
int item,front=0,rear=-1,q[10];
void insertrear()
{
    if(rear==Que_Size-1)
    {
        printf("Queue Overflow\n");
        return;
    }
    rear+=1;
    q[rear]=item;
}
int deletefront()
{
    if(front>rear)
    {
        front=0;
        rear=-1;
        return -1;
    }
    return q[front++];
}
```

```
void display()
{
    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("\n 1:Insertion\n 2:Deletion\n 3:Display\n 4:Exit\n");
        printf("Enter your choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
}

          case 1 : printf("Enter 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 : display();
                   break;
          default : exit(0);
      }

  }

}
```

OUTPUT :



```
 1:Insertion
 2:Deletion
 3:Display
 4:Exit
Enter your choice
2
Queue is empty

 1:Insertion
 2:Deletion
 3:Display
 4:Exit
Enter your choice
3
Queue is Empty

 1:Insertion
 2:Deletion
 3:Display
 4:Exit
Enter your choice
1
Enter item to be inserted
12

 1:Insertion
 2:Deletion
 3:Display
 4:Exit
Enter your choice
1
Enter item to be inserted
13

 1:Insertion
 2:Deletion
 3:Display
 4:Exit
Enter your choice
1
Enter item to be inserted
14
```



```
 1:Insertion
 2:Deletion
 3:Display
 4:Exit
Enter your choice
1
Enter item to be inserted
15
Queue Overflow

 1:Insertion
 2:Deletion
 3:Display
 4:Exit
Enter your choice
3
contents of queue
12
13
14

 1:Insertion
 2:Deletion
 3:Display
 4:Exit
Enter your choice
2
Item deleted=12

 1:Insertion
 2:Deletion
 3:Display
 4:Exit
Enter your choice
4

Process returned 0 (0x0)   execution time : 48.751 s
Press any key to continue.
```

3) **Lab4 :**

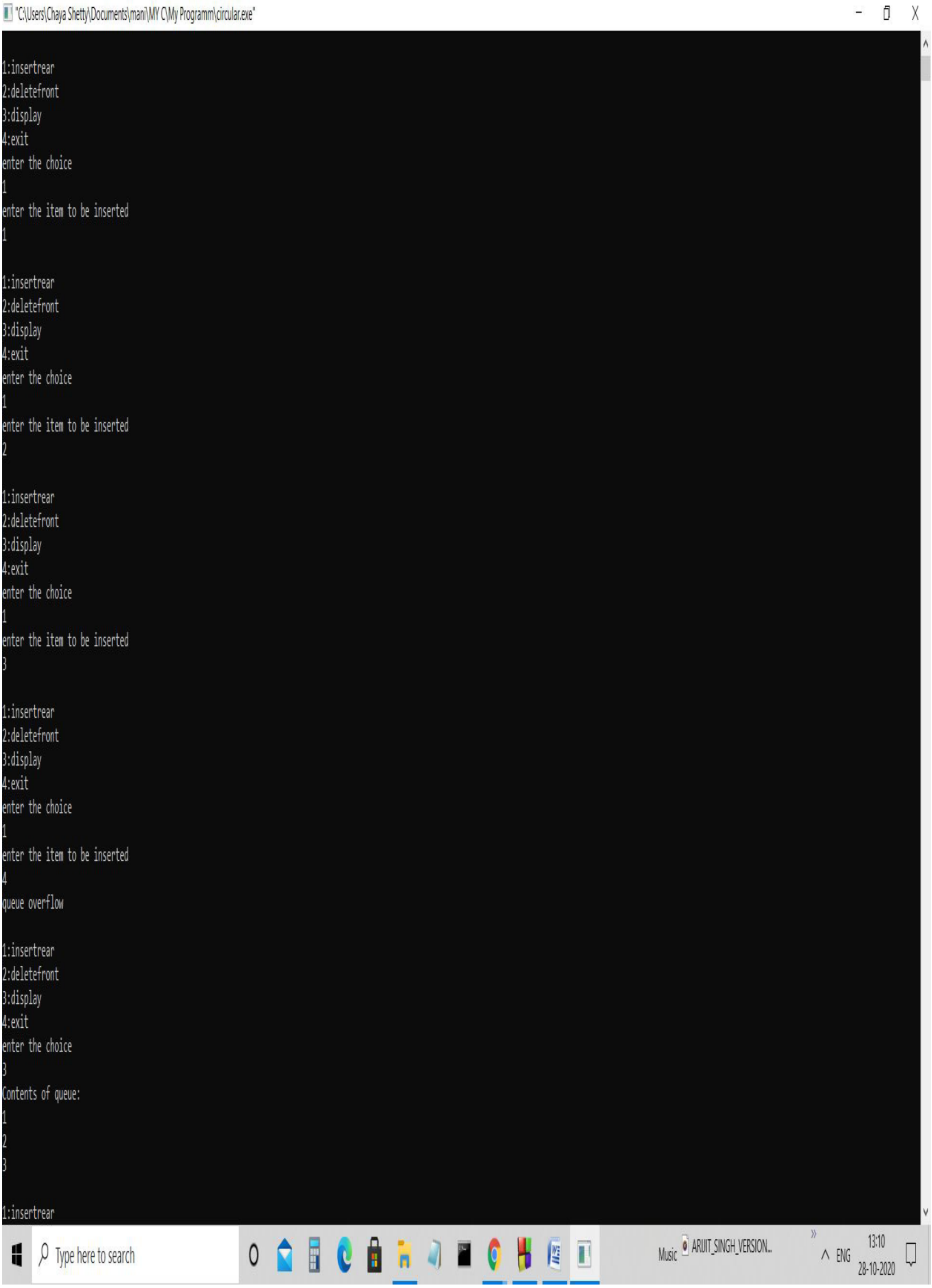
WAP to simulate the working of a circular queue of integers using an array. Provide the
following operations.
a) Insert b) Delete c) Display
The program should print appropriate messages for queue empty and queue overflow
conditions

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

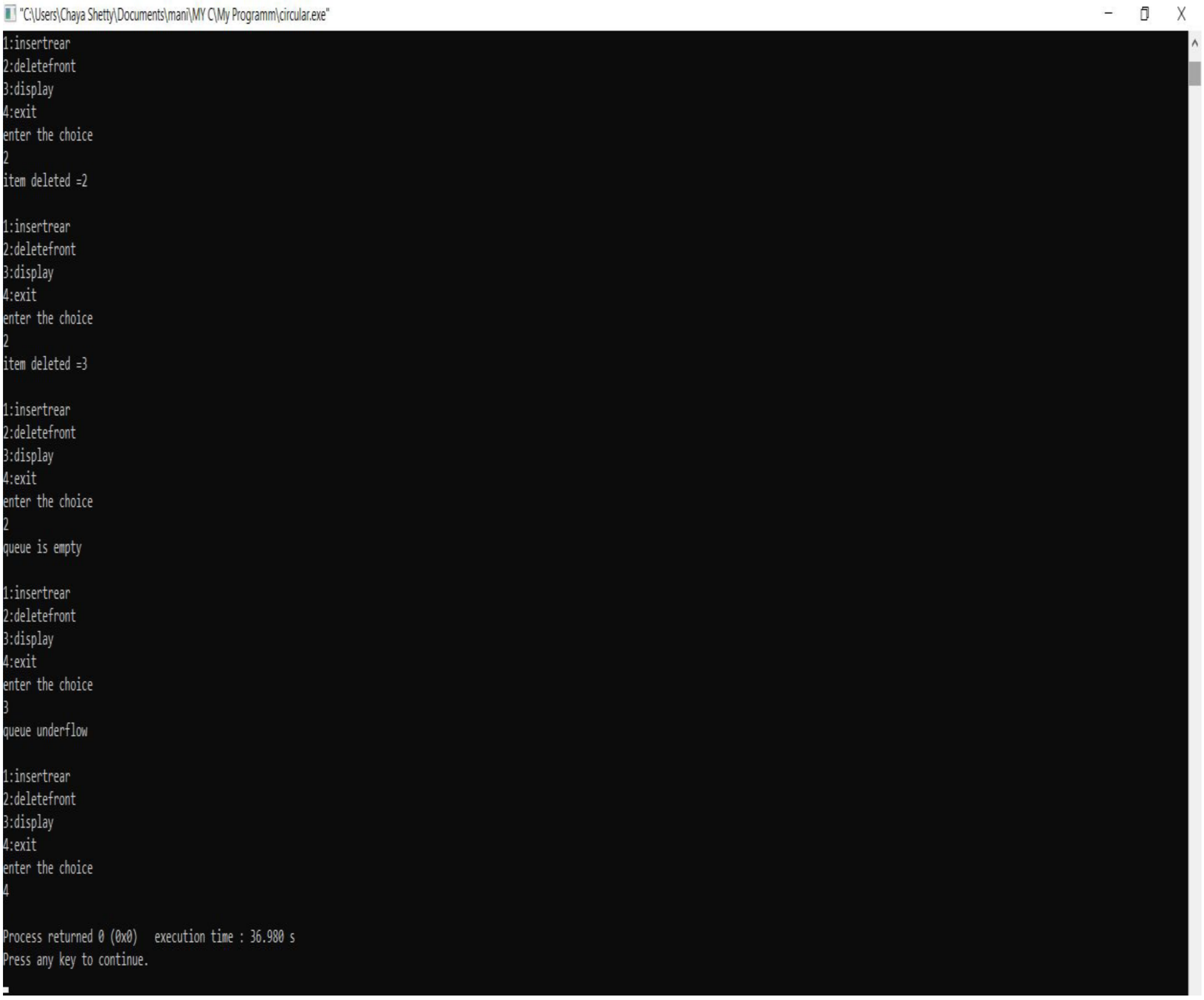
```
if(count==0)
{
printf("queue underflow\n");
return;
}
f=front;
printf("Contents of queue: \n");
for(i=1;i<=count;i++)
{
printf("%d\n",q[f]);
f=(f+1)%QUE_SIZE;
}
}
void main()
{
 int choice;



 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);

}

}

}
```

OUTPUT :


"C:\Users\Chaya Shetty\Documents\mani\MY C\My Programm\circular.exe"

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

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

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

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

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

1:insertrear
```


"C:\Users\Chaya Shetty\Documents\mani\MY C\My Programm\circular.exe"

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

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

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 underflow

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

Process returned 0 (0x0)   execution time : 36.980 s
Press any key to continue.
```

5)**Lab5 :**

WAP to Implement Singly Linked List with following operations
a) a) Create a linked list. b) Insertion of a node at first position, at any position and at end of
list. c) Display the contents of the linked list.

```
#include<stdio.h>

#include<stdlib.h>

#include<process.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)
```

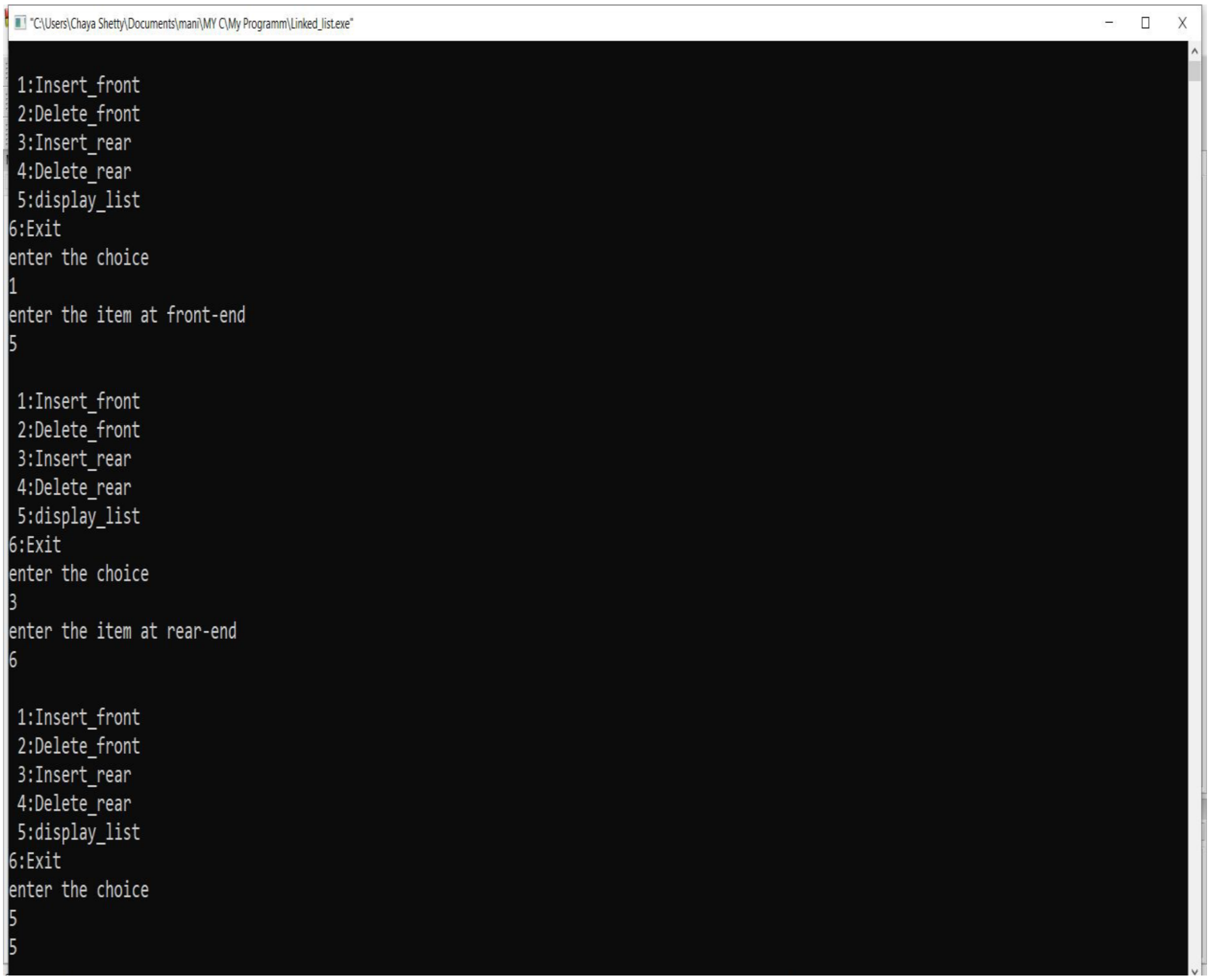
```
return temp;
temp->link=first;
first=temp;
return first;
}
NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",first->info);
free(first);
return temp;
}
NODE insert_rear(NODE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

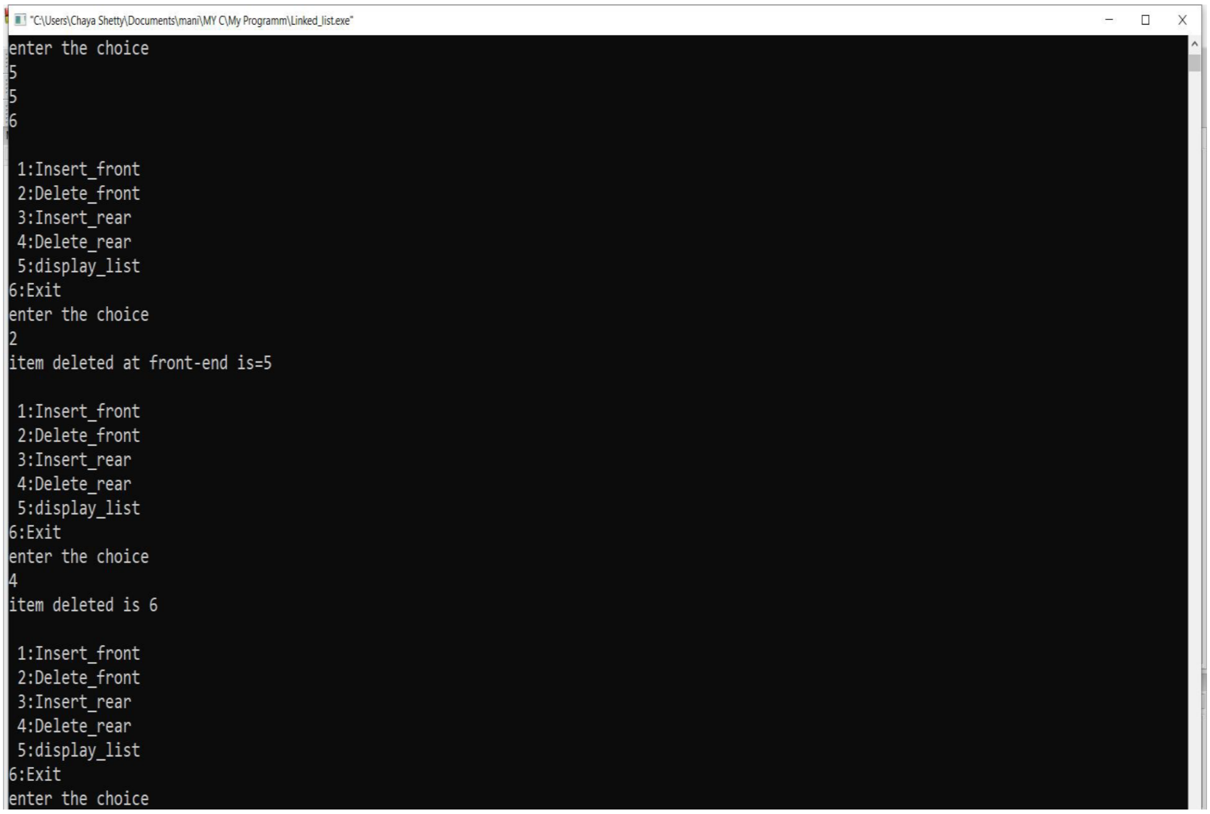
```
cur=cur->link;
}
printf("iten deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}

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);
  }
}
void main()
{
int item,choice;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n
5:display_list\n6: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:display(first);
           break;
 default:exit(0);
           break;
 }
}
}
```

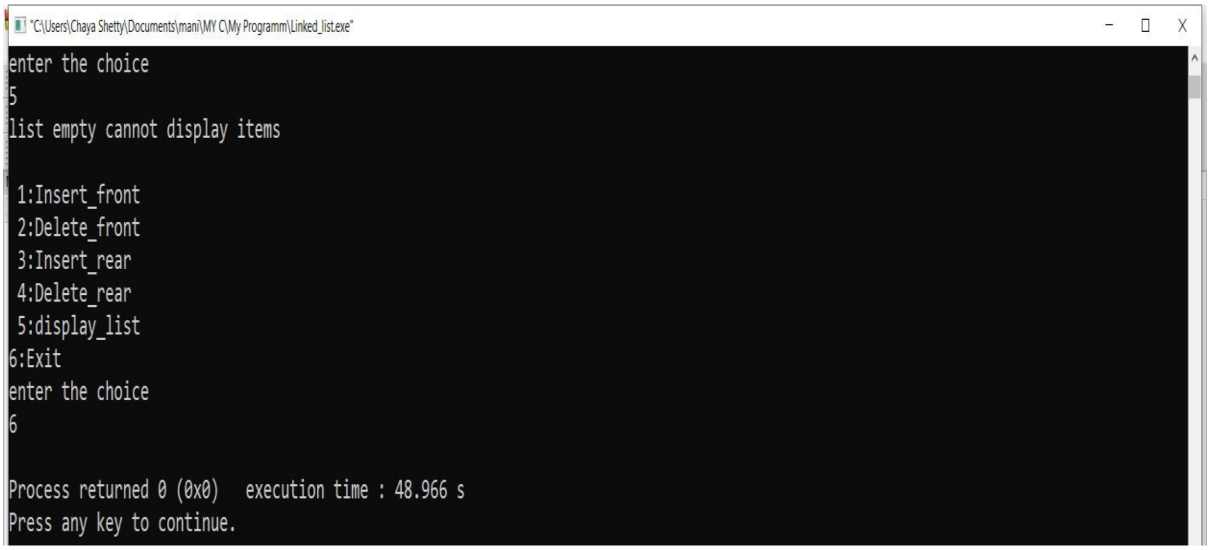
OUTPUT :


"C:\Users\Chaya Shetty\Documents\mani\MY C\My Programm\Linked_list.exe"

```

 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
5

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:display_list
6:Exit
enter the choice
3
enter the item at rear-end
6

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


"C:\Users\Chaya Shetty\Documents\mani\MY C\My Programm\Linked_list.exe"

```
enter the choice
5
5
6

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:display_list
6:Exit
enter the choice
2
item deleted at front-end is=5

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:display_list
6:Exit
enter the choice
4
item deleted is 6

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



"C:\Users\Chaya Shetty\Documents\mani\MY C\My Programm\Linked_list.exe"

```
enter the choice
5
list empty cannot display items

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:display_list
6:Exit
enter the choice
6

Process returned 0 (0x0)   execution time : 48.966 s
Press any key to continue.
```

6)**Lab6 :**

WAP to Implement Singly Linked List with following operations
a) a) Create a linked list. b) Deletion of first element, specified element and last element in
the list. c) Display the contents of the linked list.

```
#include<stdio.h>

#include<stdlib.h>

#include<process.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;
```

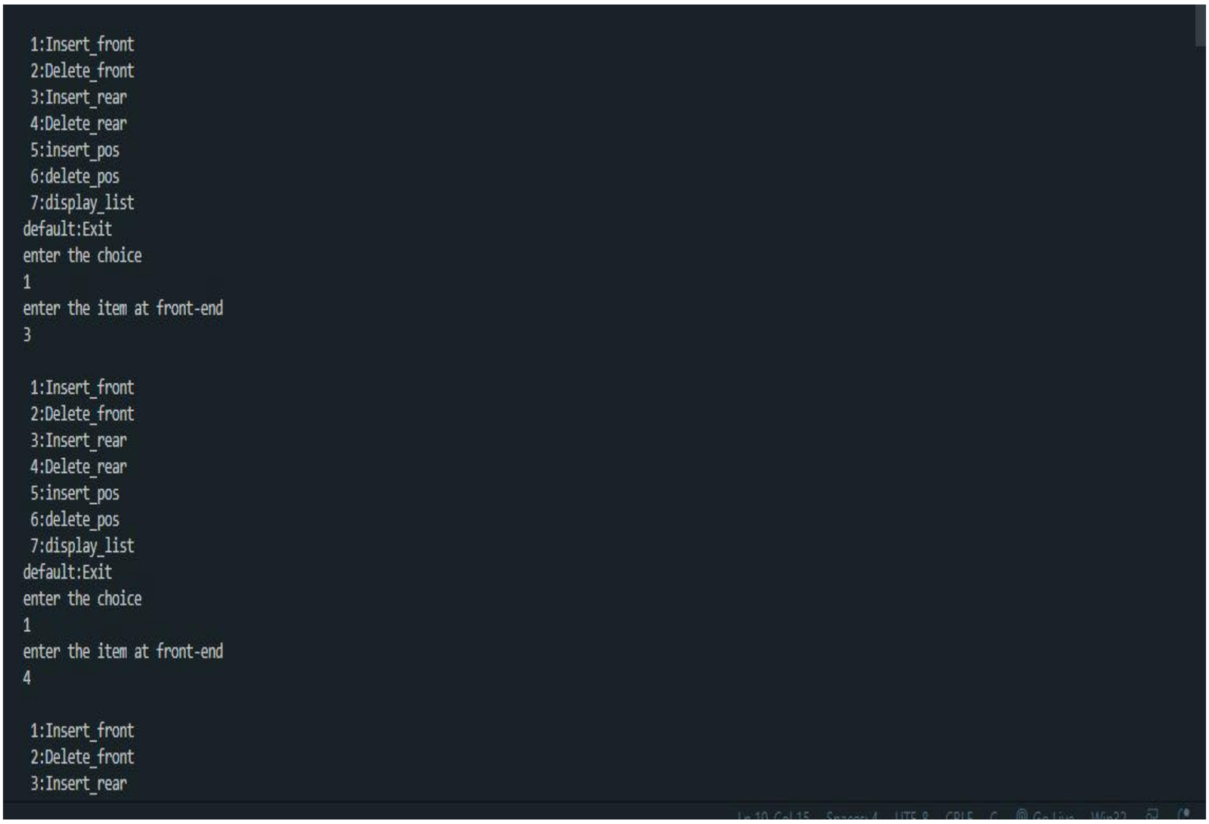
```
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
if(first->link==NULL)
{
printf("item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
printf("iten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

```
}
NODE delete_info(int key,NODE first)
{
NODE prev,cur;
if(first==NULL)
{
printf("list is empty\n");
return NULL;
}
if(key==first->info)
{
cur=first;
first=first->link;
freenode(cur);
return first;
}
prev=NULL;
cur=first;
while(cur!=NULL)
{
if(key==cur->info)break;
prev=cur;
cur=cur->link;
}
if(cur==NULL)
{
```

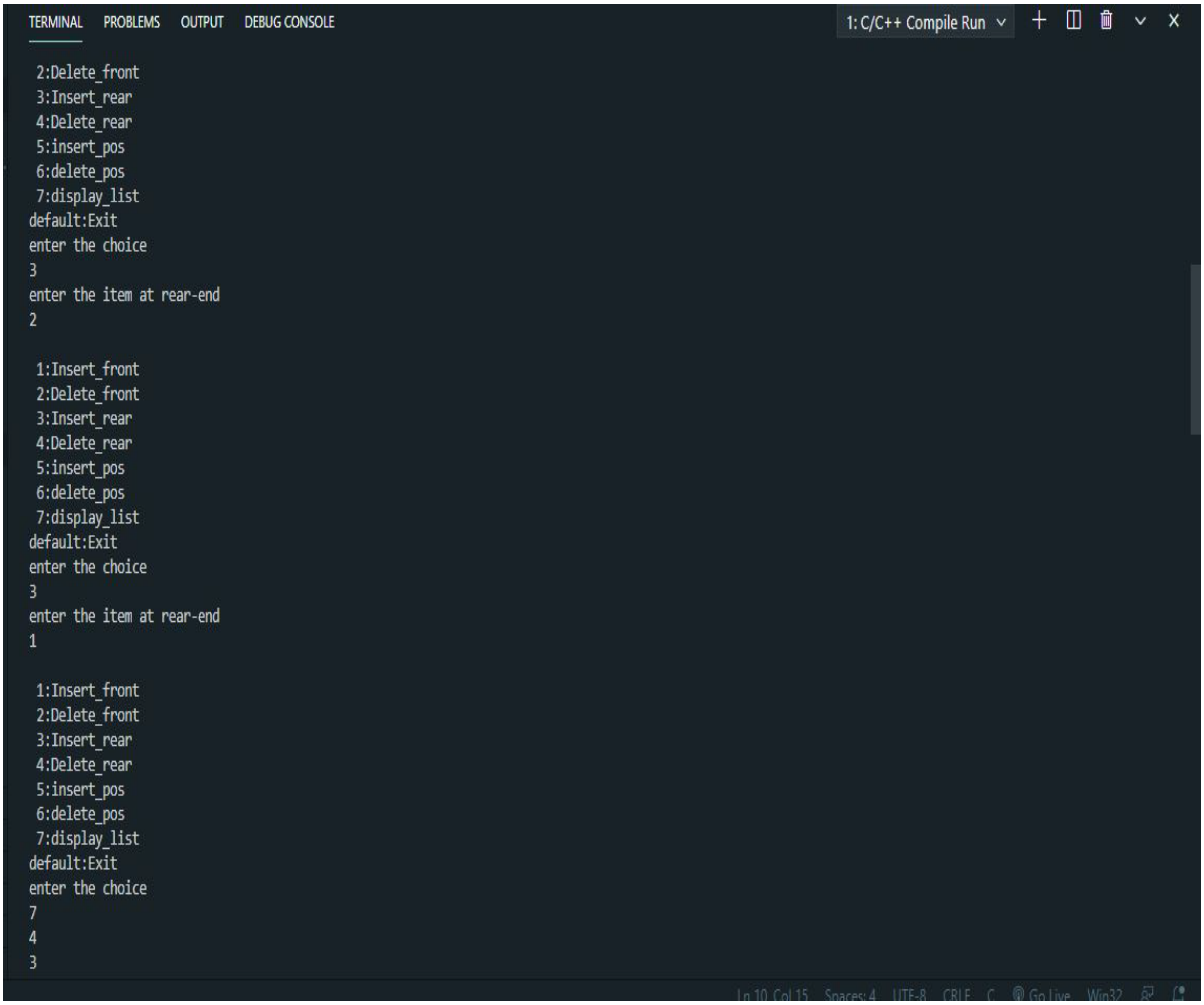
```
printf("search is unsuccessfull\n");
return first;
}
prev->link=cur->link;
printf("key deleted is %d",cur->info);
freenode(cur);
return first;
}
NODE insert_pos(int item,int pos,NODE first)
{
        NODE temp,cur,prev;
        int count;
        temp=getnode();
        temp->info=item;
        temp->link=NULL;
        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;
}
void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("list empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\n",temp->info);
  }
}
void main()
{
int item,choice,pos,key;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n
5:delete_pos\n 6:insert_pos\n 7:display_list\n8: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 key to be deleted\n");
                scanf("%d",&key);
                first=delete_info(key,first);
                break;
  case 6:printf("Enter the item and the position:\n");
         scanf("%d%d",&item,&pos);
         first=insert_pos(item,pos,first);
         break;
  case 7:display(first);
            break;

 default:exit(0);


     break;


 }


}


}
```

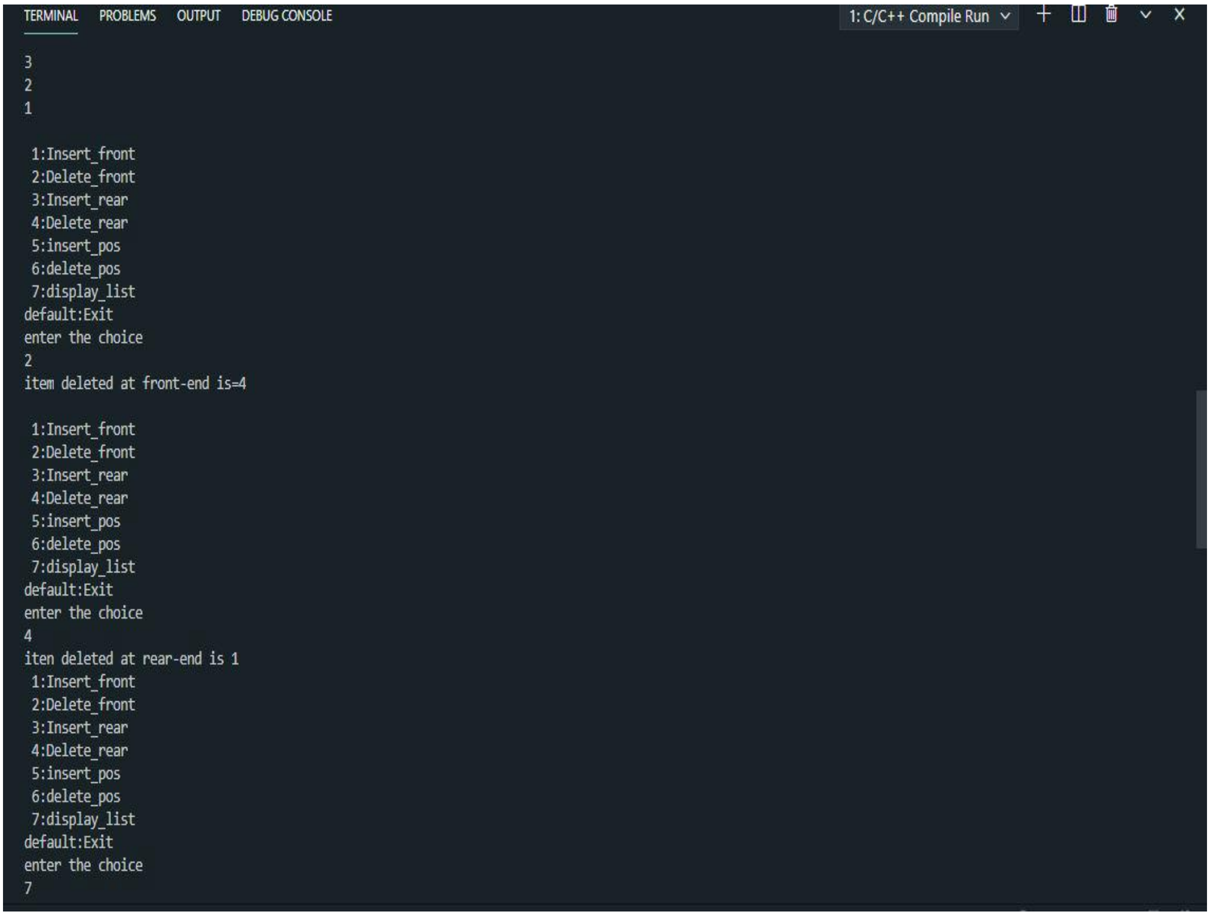
OUTPUT :



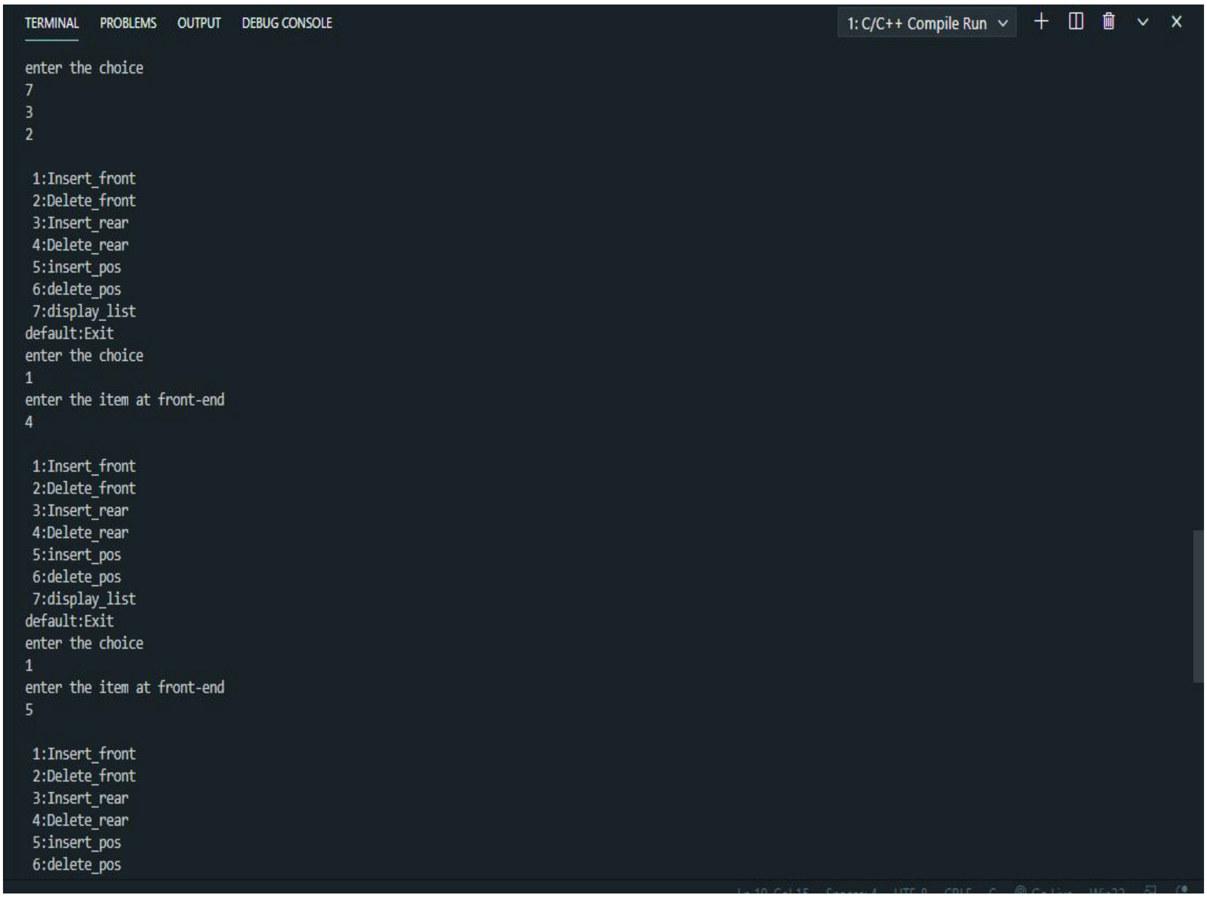
```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
1
enter the item at front-end
3

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
1
enter the item at front-end
4

 1:Insert_front
 2:Delete_front
 3:Insert_rear
```



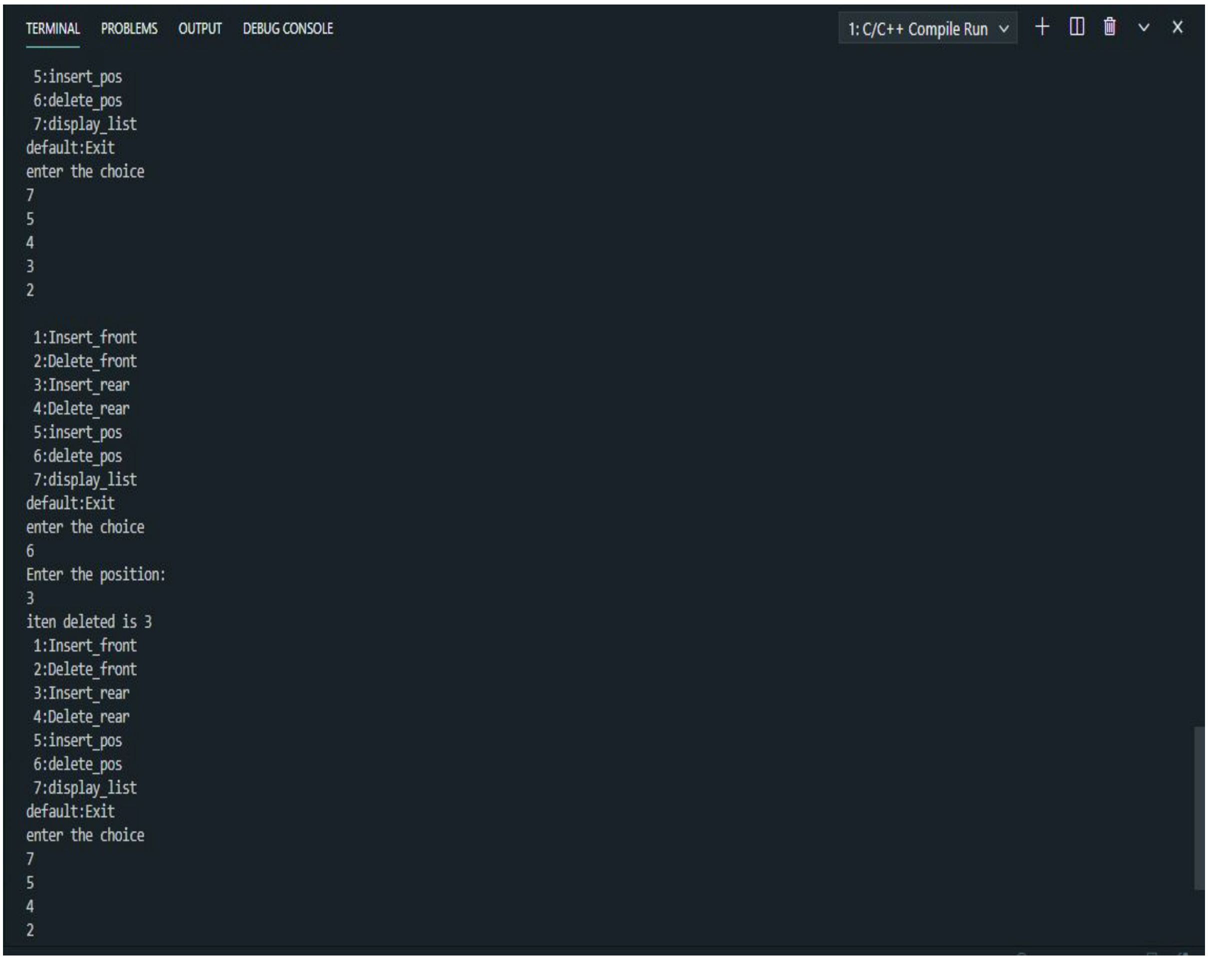
```
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
3
enter the item at rear-end
2

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
3
enter the item at rear-end
1

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
7
4
3
```



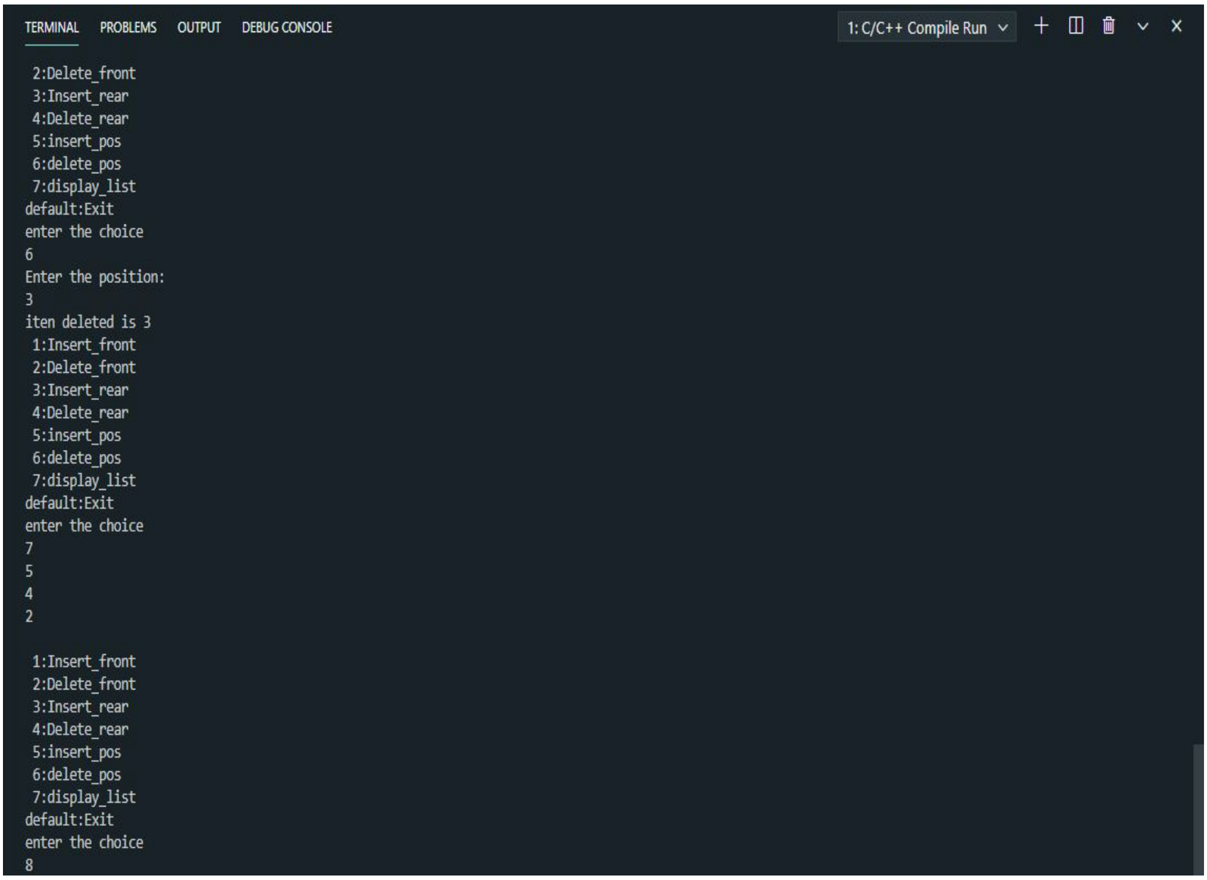
```
3
2
1

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
2
item deleted at front-end is=4

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
4
iten deleted at rear-end is 1
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
7
```



```
enter the choice
7
3
2

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
1
enter the item at front-end
4

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
1
enter the item at front-end
5

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
```



```
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
7
5
4
3
2

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
6
Enter the position:
3
iten deleted is 3
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
7
5
4
2
```



```
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
6
Enter the position:
3
iten deleted is 3
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
7
5
4
2

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:insert_pos
 6:delete_pos
 7:display_list
default:Exit
enter the choice
8
```

7)Lab7 :

WAP Implement Single Link List with following operations
a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists

```
#include<stdio.h>
#include<stdlib.h>
#include<process.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;
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
```

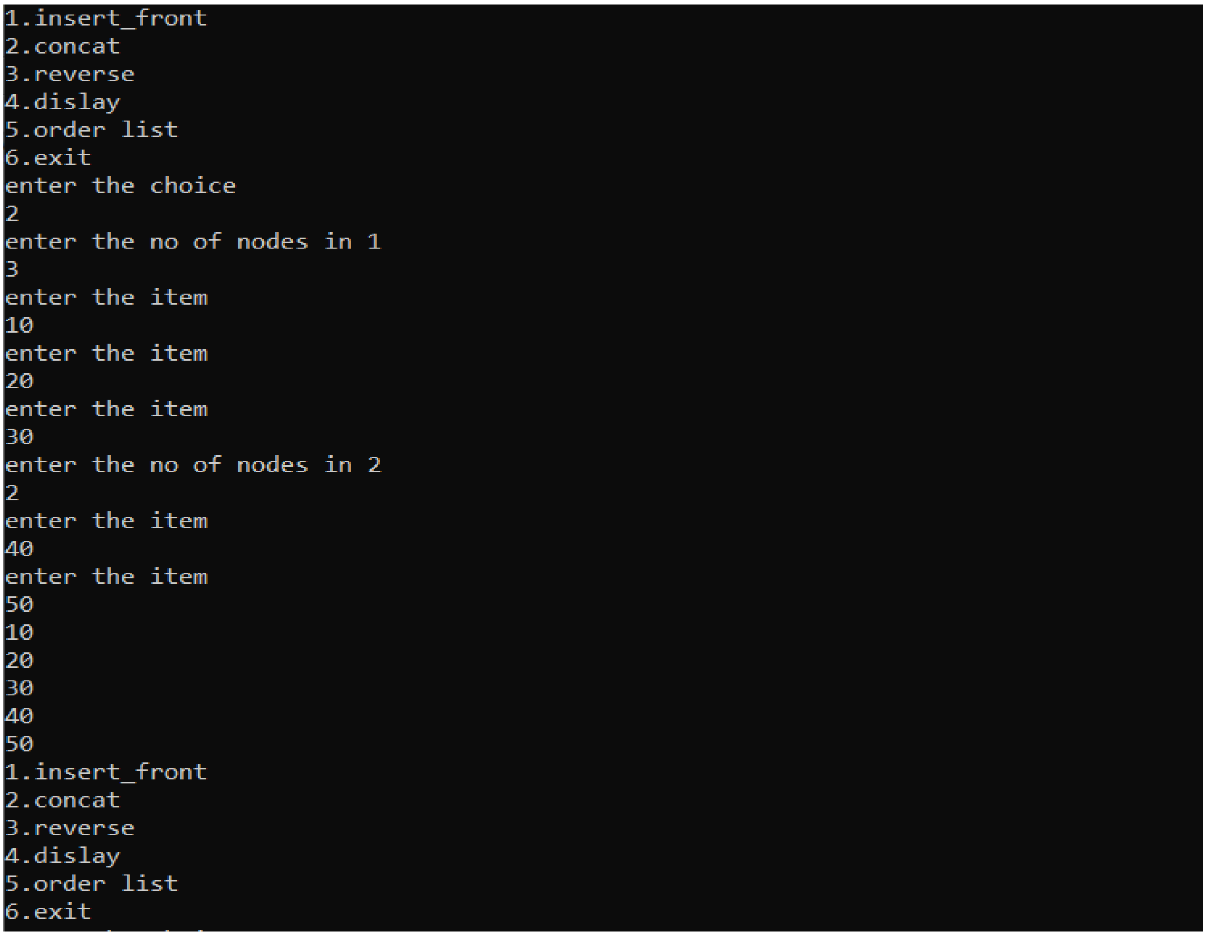
```
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
 return temp;
cur=first;
while(cur->link!=NULL)
 cur=cur->link;
cur->link=temp;
return first;
}
void display(NODE first)
{
 NODE temp;
 if(first==NULL)
  printf("list empty");

 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(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;
}

void main()
{
int item,choice,pos,i,n;
NODE first=NULL,a,b;

for(;;)
{
printf("1.insert_front\n2.concat\n3.reverse\n4.dislay\n5.order list\n6.exit\n");
printf("enter the choice\n");
```

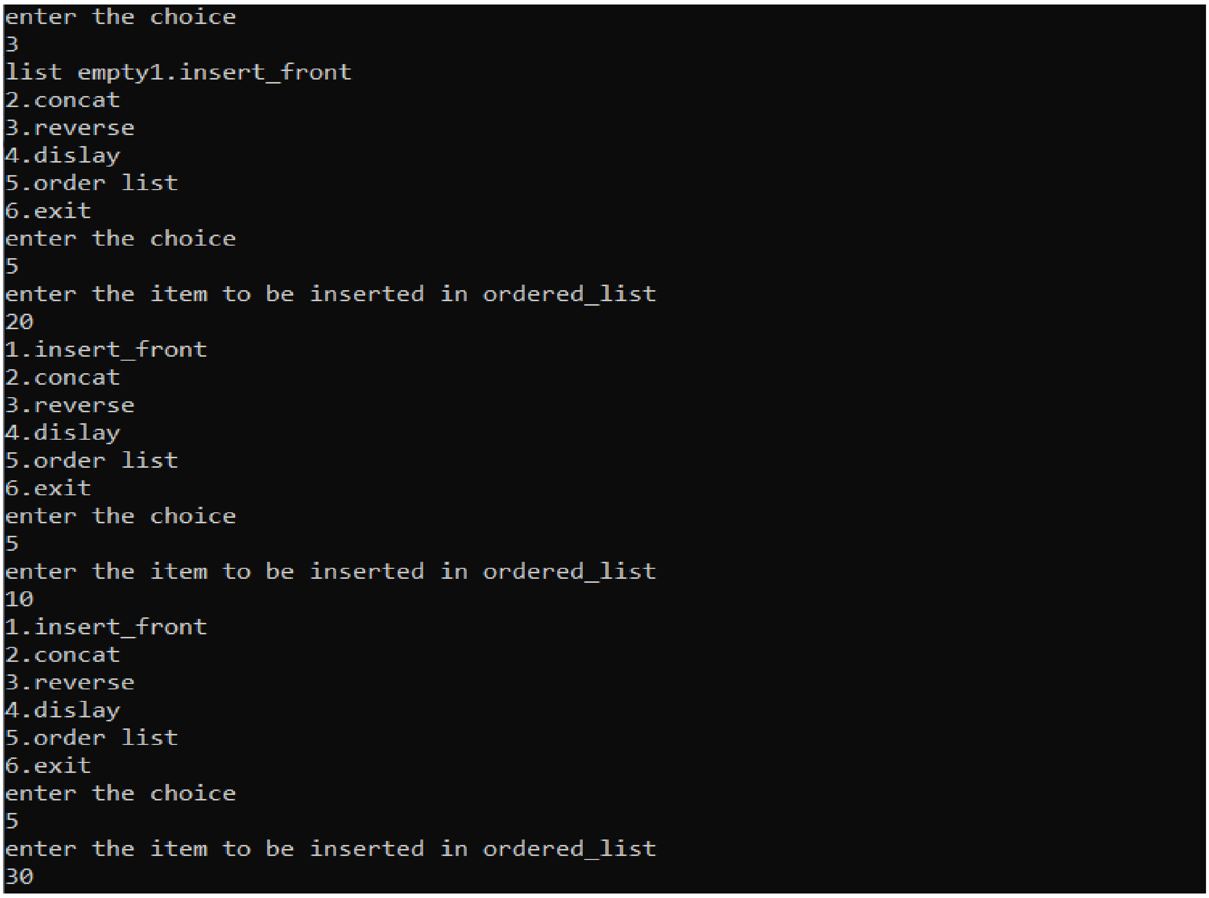
```
scanf("%d",&choice);

switch(choice)
 {

case 1:printf("enter the item\n");
                  scanf("%d",&item);
                  first=insert_rear(first,item);
                  break;

case 2: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 3:first=reverse(first);
                    display(first);
                    break;

case 4:display(first);
                    break;

case 5:printf("enter the item to be inserted in ordered_list\n");
               scanf("%d",&item);
          first=order_list(item,first);
              break;

  default:exit(0);

}

 }

  }
```

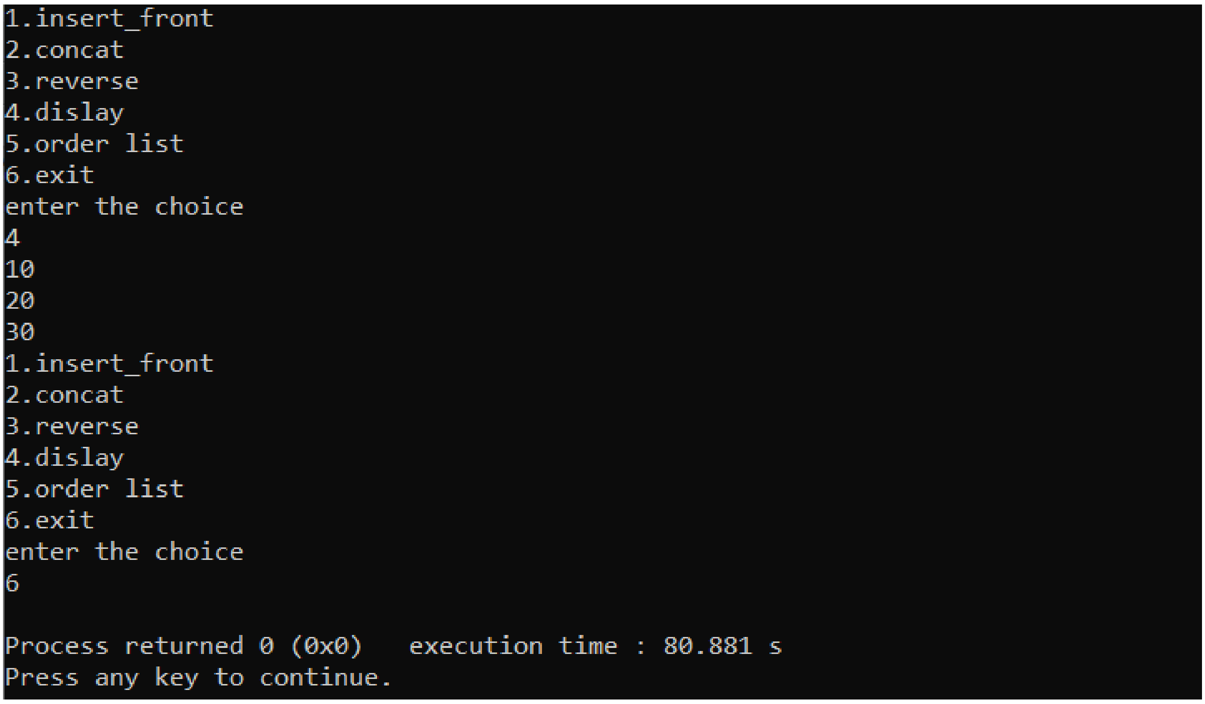
OUTPUT :



```
1.insert_front
2.concat
3.reverse
4.dislay
5.order list
6.exit
enter the choice
2
enter the no of nodes in 1
3
enter the item
10
enter the item
20
enter the item
30
enter the no of nodes in 2
2
enter the item
40
enter the item
50
10
20
30
40
50
1.insert_front
2.concat
3.reverse
4.dislay
5.order list
6.exit
```



```
enter the choice
3
list empty1.insert_front
2.concat
3.reverse
4.dislay
5.order list
6.exit
enter the choice
5
enter the item to be inserted in ordered_list
20
1.insert_front
2.concat
3.reverse
4.dislay
5.order list
6.exit
enter the choice
5
enter the item to be inserted in ordered_list
10
1.insert_front
2.concat
3.reverse
4.dislay
5.order list
6.exit
enter the choice
5
enter the item to be inserted in ordered_list
30
```



```
1.insert_front
2.concat
3.reverse
4.dislay
5.order list
6.exit
enter the choice
4
10
20
30
1.insert_front
2.concat
3.reverse
4.dislay
5.order list
6.exit
enter the choice
6

Process returned 0 (0x0)   execution time : 80.881 s
Press any key to continue.
```

8)**Lab8 :**

WAP to implement Stack & Queues using Linked Representation

Stack :

```
#include<stdio.h>

#include<stdlib.h>

#include<process.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("stack 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("stack empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
}
void main()
{
```

```
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_front\n 2:Delete_front\n 3:Display_list\n 4: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:display(first);
break;
default:exit(0);
break;
}
}
}
```

Queue :

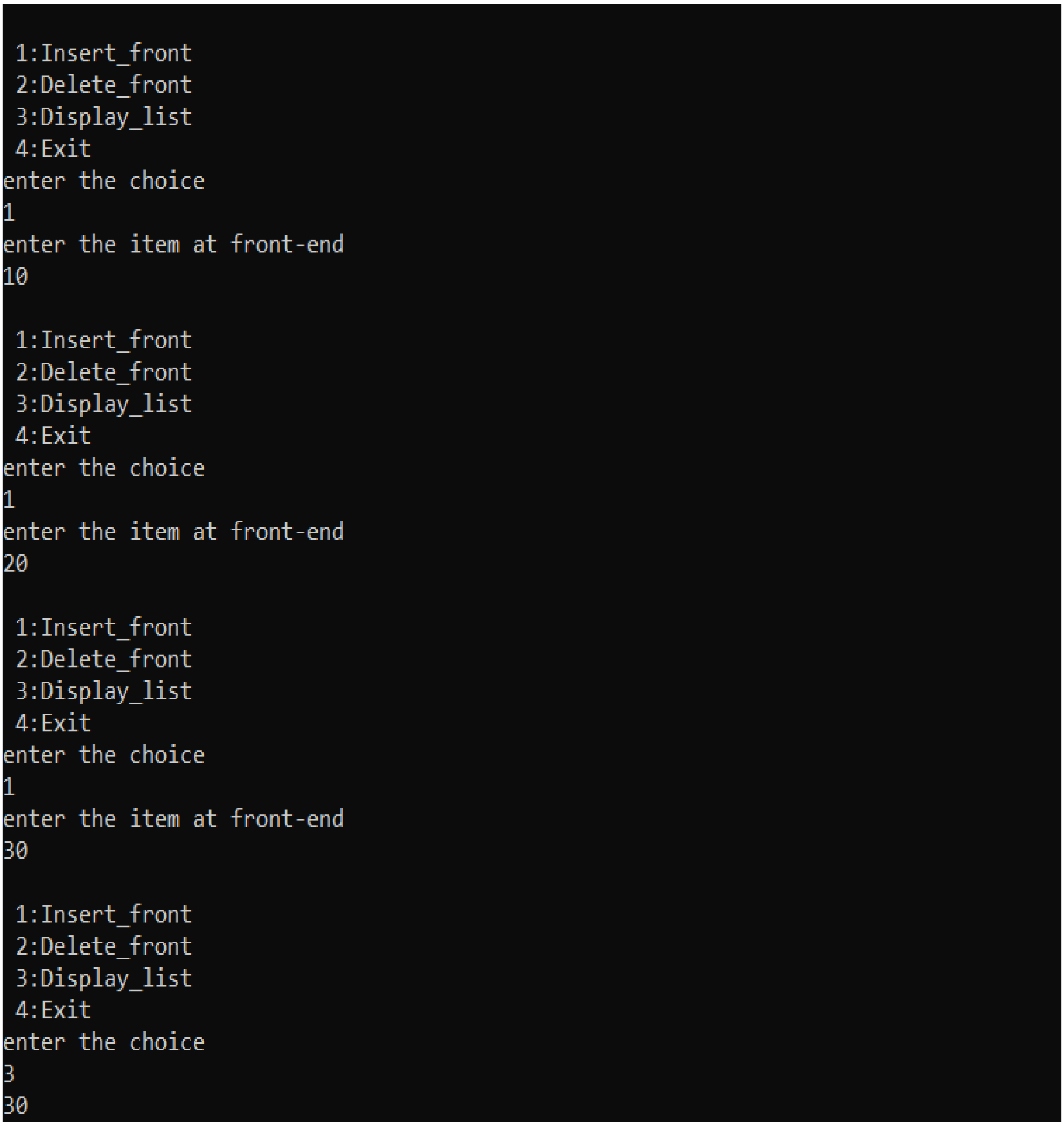
```
#include<stdio.h>
#include<stdlib.h>
```

```
#include<process.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();
```

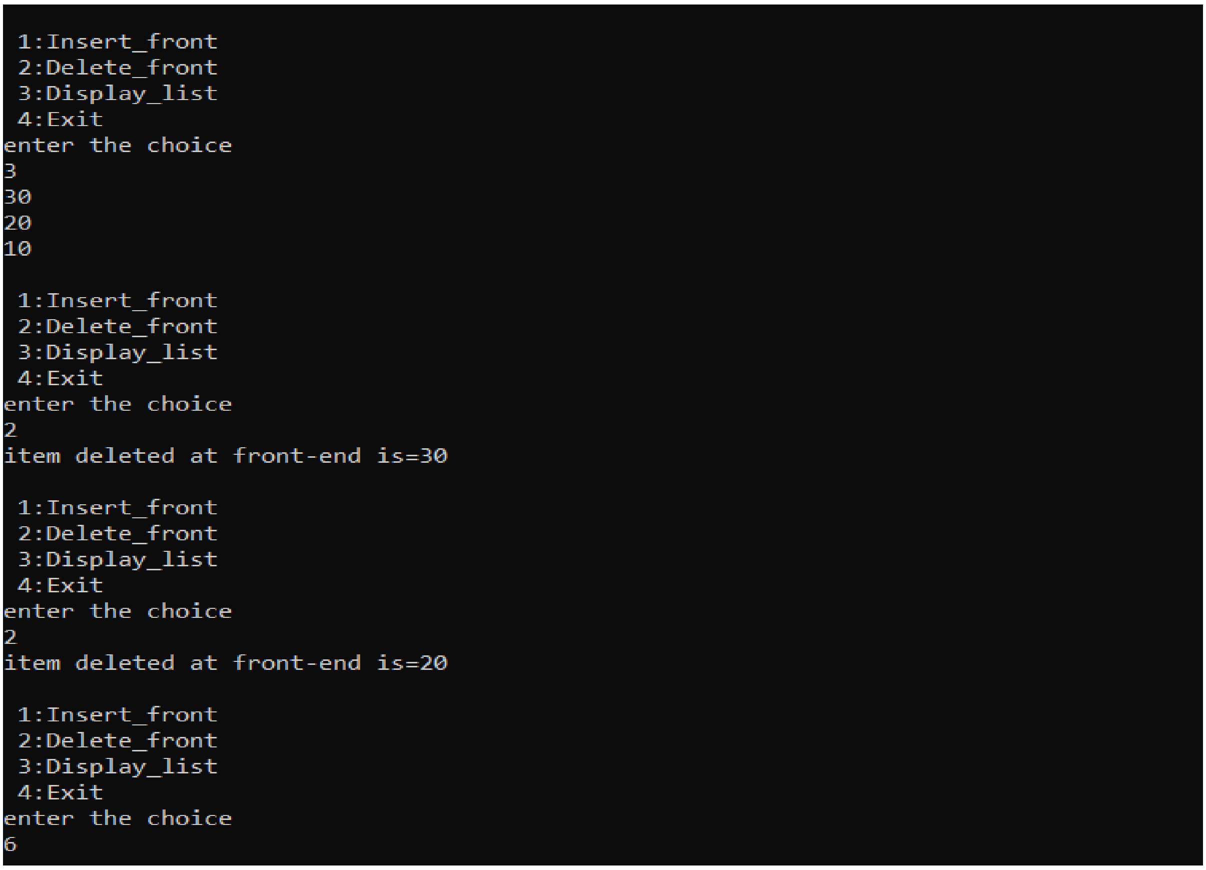
```
temp->info=item;

temp->link=NULL;

if(first==NULL)

return temp;

cur=first;

while(cur->link!=NULL)

cur=cur->link;

cur->link=temp;

return first;

}

NODE delete_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);
}
}
void main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_rear\n 2:Delete_front\n 3:Display_list\n 4:Exit\n");
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_front(first);
break;case 3:display(first);
break;
default:exit(0);

break;

}

}

}
```

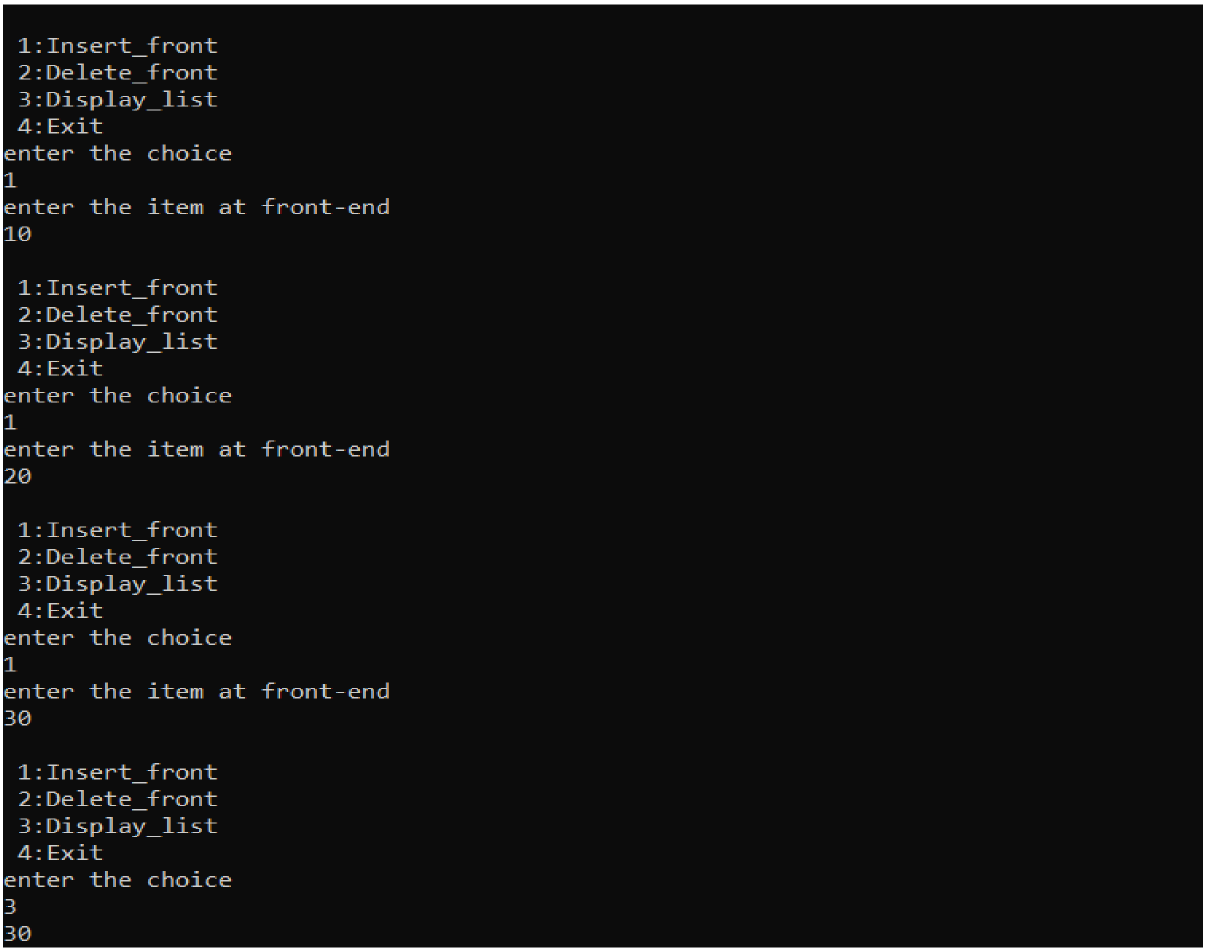
OUTPUT :



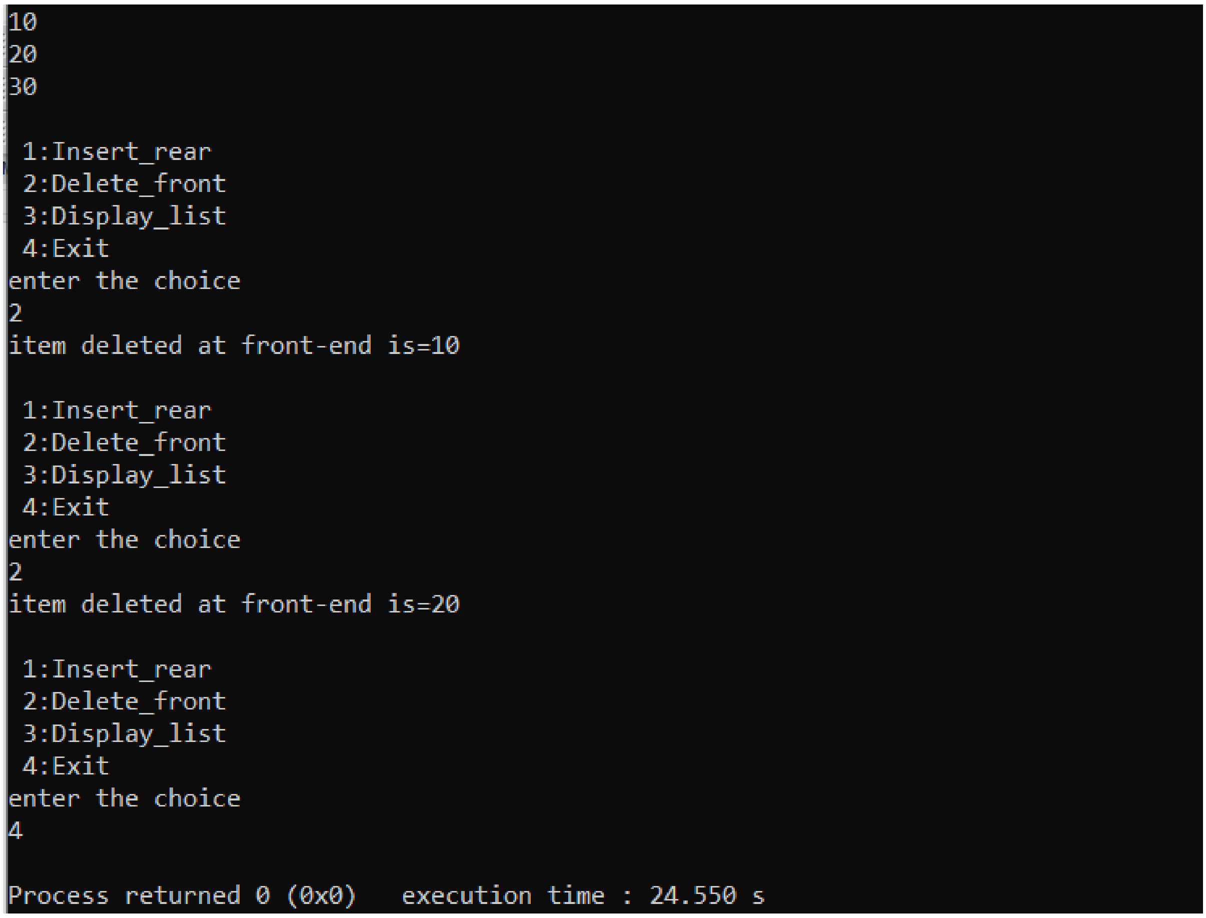
```

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at front-end
10

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at front-end
20

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at front-end
30

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
3
30
```



```
 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
3
30
20
10

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

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
item deleted at front-end is=20

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
6
```



```
 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at front-end
10

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at front-end
20

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
1
enter the item at front-end
30

 1:Insert_front
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
3
30
```



```
10
20
30

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
item deleted at front-end is=10

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
enter the choice
2
item deleted at front-end is=20

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

Process returned 0 (0x0)   execution time : 24.550 s
```

9)**Lab9 :**

WAP Implement doubly link list with primitive operations
a) a) Create a doubly linked list. b) Insert a new node to the left of the node.
b) c) Delete the node based on a specific value. c) Display the contents of the list

Part-1:

```
#include<stdio.h>

#include<stdlib.h>

#include<process.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("dq empty\n");
return head;
}
```

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

```
NODE temp;
if(head->rlink==head)
{
printf("dq empty\n");
return;
}
printf("contents of dq\n");
temp=head->rlink;
while(temp!=head)
{
printf("%d\n",temp->info);
temp=temp->rlink;
}
printf("\n");
}
void main()
{
NODE head,last;
int item, choice;
head=getnode();
head->rlink=head;
head->llink=head;
for(;;)
{
        printf("\n1:insert front\n2:insert rear\n3:delete front\n4:delete
rear\n5:display\n6: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);
                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: display(head);
                break;

           default:exit(0);

    }

    }

    }
```

Part-2:

```
#include<stdio.h>
#include<stdlib.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_rear(NODE head,int item)
{
NODE temp,cur;
temp=getnode();
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
cur=head->llink;
temp->llink=cur;
cur->rlink=temp;
head->llink=temp;
temp->rlink=head;
head->info=head->info+1;
return head;
}
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_righttpos(int item,NODE head)
{
NODE temp,cur,prev;
if(head->rlink==head)
{
printf("list empty\n");
return head;
}
```

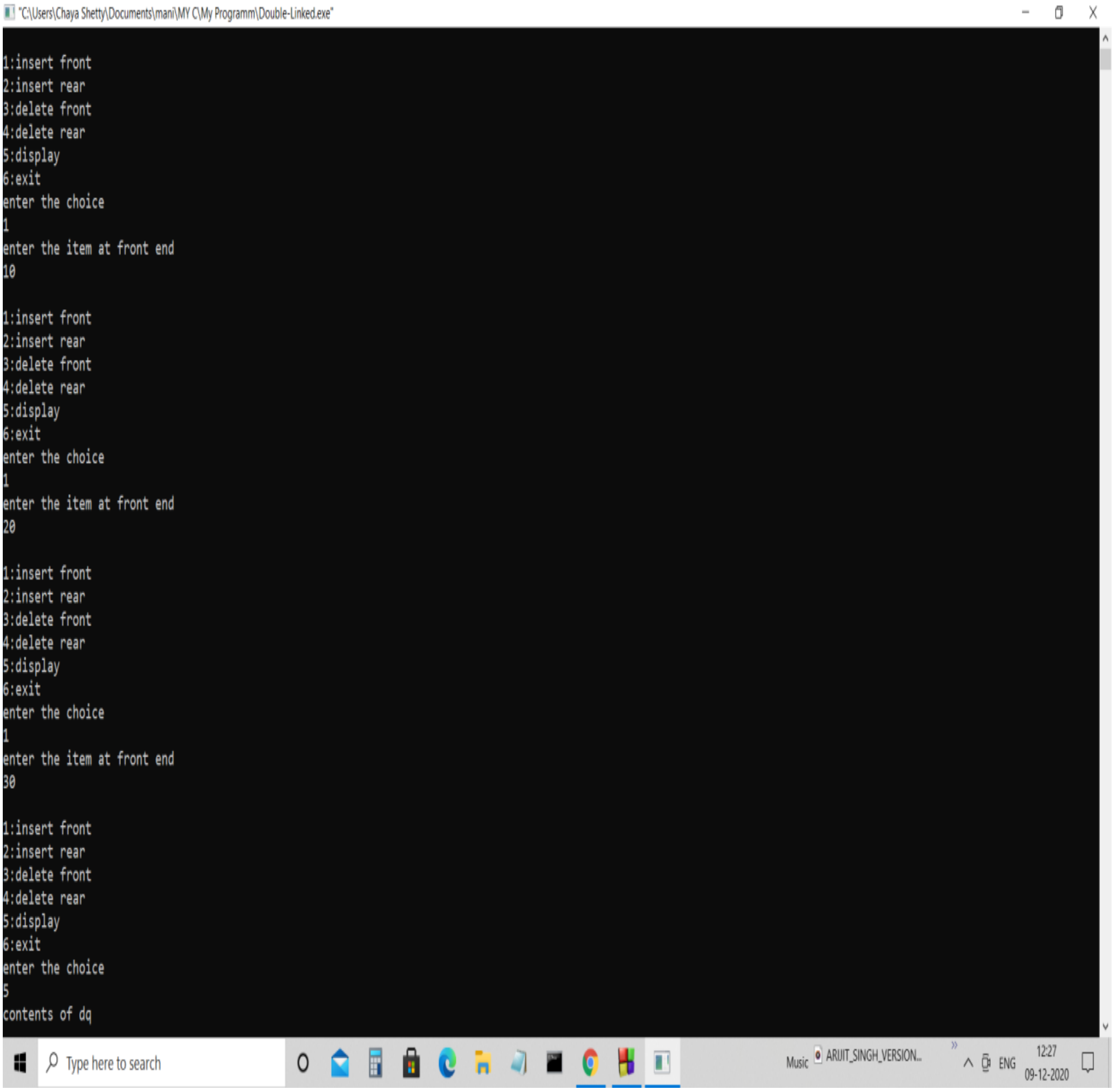
```
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)break;
cur=cur->rlink;
}
if(cur==head)
{
 printf("key not found\n");
 return head;
 }
 prev=cur->rlink;
 printf("enter towards left of %d=",item);
 temp=getnode();
 scanf("%d",&temp->info);
 prev->llink=temp;
 temp->llink=cur;
 cur->rlink=temp;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

```
    {
     printf("LE");
     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);
  cur=next;
 }
}
if(count==0)
  printf("key not found");
else
  printf("key found at %d positions and are deleted\n", count);
```

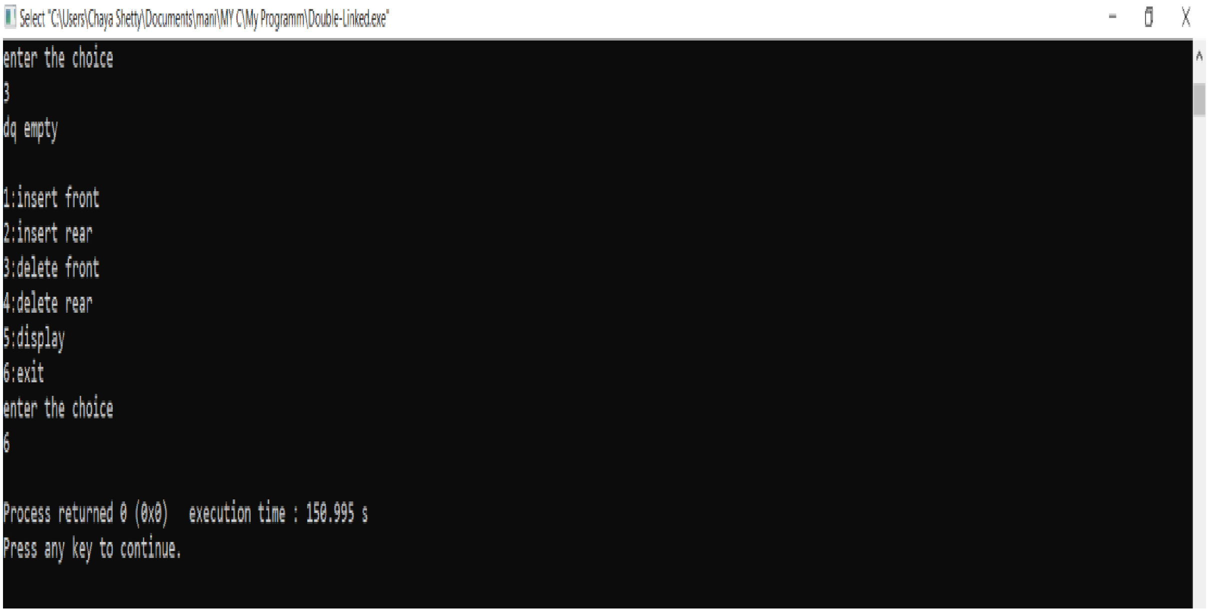
```
return head;
}
void Search_info(int item,NODE head){
NODE cur;
if(head->rlink==head)
{
printf("list empty\n");
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)
{
   printf("Search Successfull\n");
   break;
}
cur=cur->rlink;
}
if(cur==head)
{
 printf("Info not found\n");
 }
}
void display(NODE head)
{
NODE temp;
```

```
if(head->rlink==head)
{
printf("list empty\n");
return;
}
for(temp=head->rlink;temp!=head;temp=temp->rlink)
printf("%d\n",temp->info);
}
void main()
{
int item,choice,key;
NODE head;
head=getnode();
head->rlink=head;
head->llink=head;
for(;;)
{
printf("\n1.insert_rear\n2.insert_key_left\n3.insert_key_right\n4.delete_duplicates\n5.Searh_info\n6.display\n7.exit\n");
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item\n");
                scanf("%d",&item);
                head=insert_rear(head,item);
                break;
  case 2:printf("enter the key item\n");
                scanf("%d",&item);
                head=insert_leftpos(item,head);
  case 3:printf("enter the key item\n");
                scanf("%d",&item);
                head=insert_righttpos(item,head);
  case 4:printf("enter the key item\n");
                scanf("%d",&item);
                head=delete_all_key(item,head);
                break;
  case 5:printf("enter the key item\n");
                scanf("%d",&item);
                Search_info(item,head);
                break;
  case 6:display(head);
                break;
  default:exit(0);
                     break;

}

}

}
```

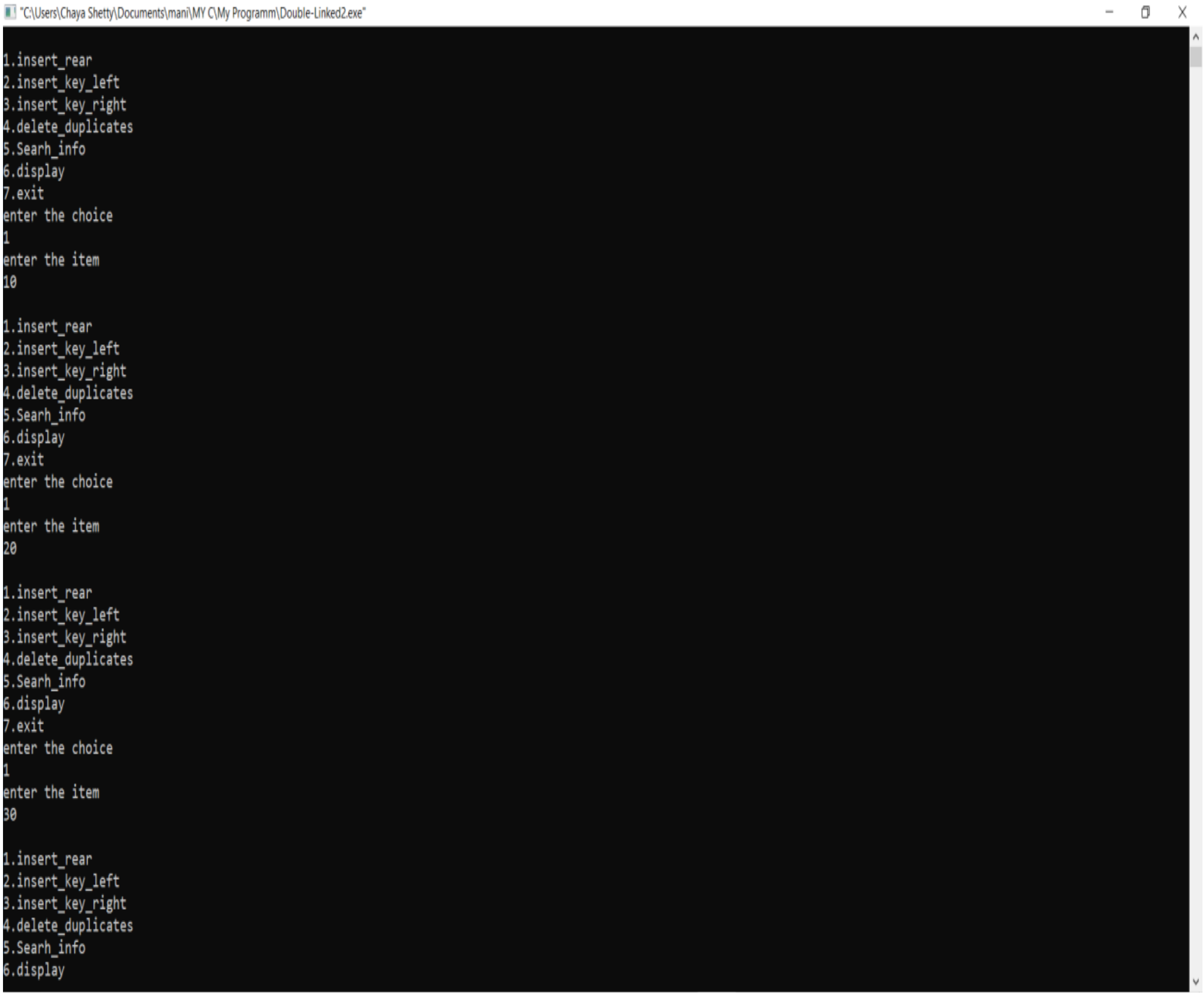
OUTPUT :


"C:\Users\Chaya Shetty\Documents\mani\MY C\My Programm\Double-Linked.exe"
Type here to search
Music
ARIJIT_SINGH_VERSION...
ENG
12:27
09-12-2020

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit
enter the choice
1
enter the item at front end
10

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit
enter the choice
1
enter the item at front end
20

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit
enter the choice
1
enter the item at front end
30

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit
enter the choice
5
contents of dq
```

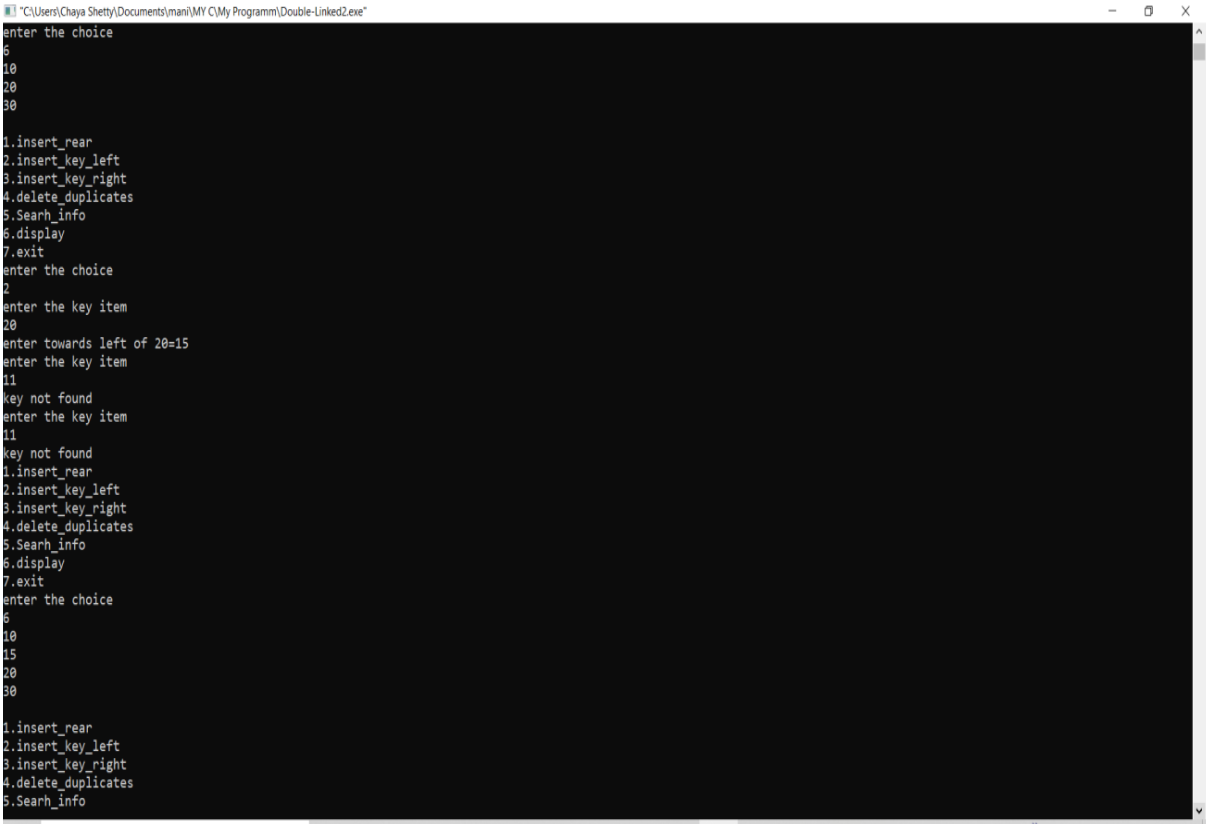

Select "C:\Users\Chaya Shetty\Documents\mani\MY C\My Programm\Double-Linked.exe"

```
enter the choice
3
dq empty

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:exit
enter the choice
6


Process returned 0 (0x0)   execution time : 150.995 s
Press any key to continue.
```



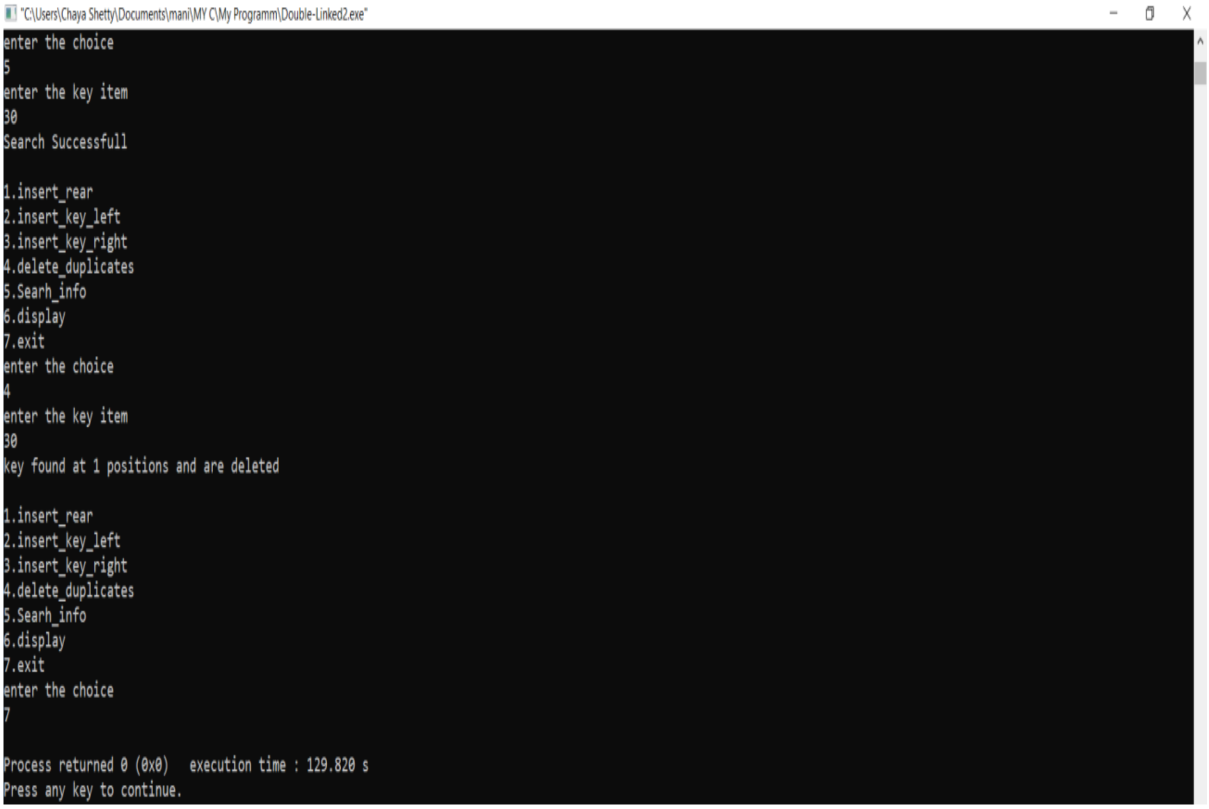
"C:\Users\Chaya Shetty\Documents\mani\MY C\My Programm\Double-Linked2.exe"

```
1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
10

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
20

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
1
enter the item
30

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
```



"C:\Users\Chaya Shetty\Documents\mani\MY C\My Programm\Double-Linked2.exe"

```
enter the choice
6
10
20
30

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
2
enter the key item
20
enter towards left of 20=15
enter the key item
11
key not found
enter the key item
11
key not found
1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
6
10
15
20
30

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
```


"C:\Users\Chaya Shetty\Documents\mani\MY C\My Programm\Double-Linked2.exe"

```
enter the choice
5
enter the key item
30
Search Successfull

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
4
enter the key item
30
key found at 1 positions and are deleted

1.insert_rear
2.insert_key_left
3.insert_key_right
4.delete_duplicates
5.Searh_info
6.display
7.exit
enter the choice
7

Process returned 0 (0x0)   execution time : 129.820 s
Press any key to continue.
```

10)**Lab10 :**

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<stdlib.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)
```

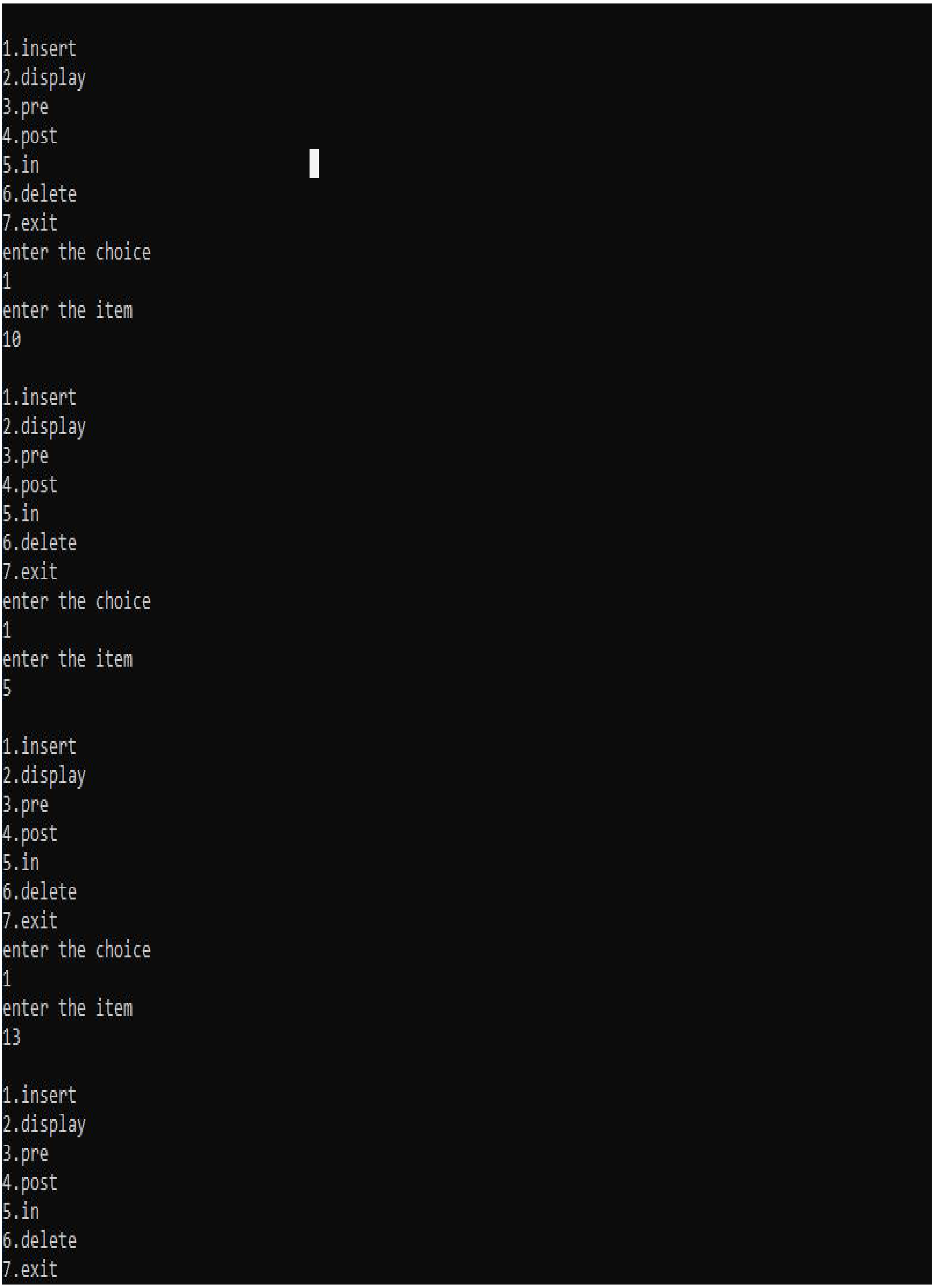
```
 return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
 prev->llink=temp;
else
 prev->rlink=temp;
return root;
}
void display(NODE root,int i)
{
int j;
if(root!=NULL)
 {
  display(root->rlink,i+1);
  for(j=0;j<i;j++)
    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

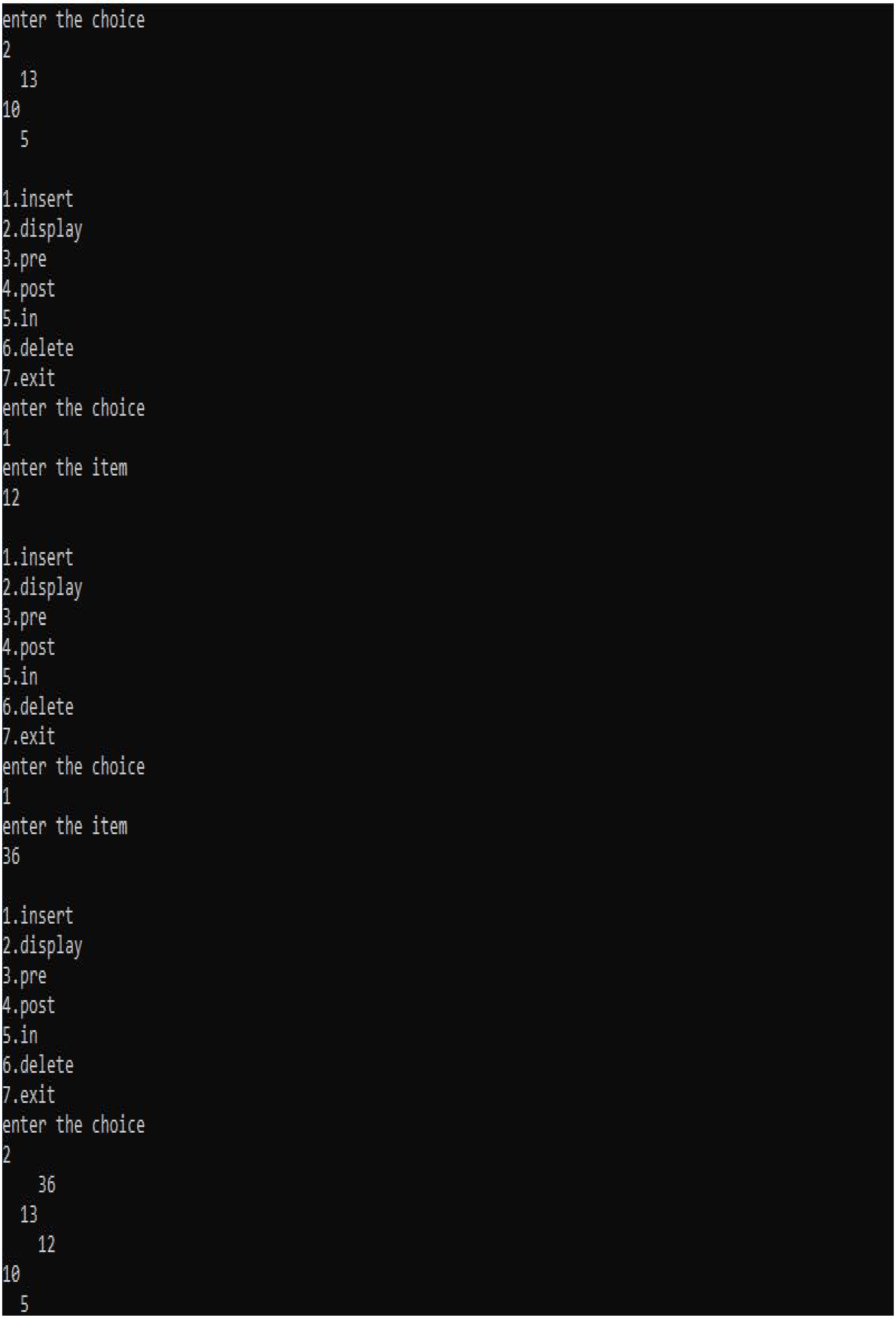
```
 suc=cur->rlink;
 while(suc->llink!=NULL)
  suc=suc->llink;
 suc->llink=cur->llink;
 q=cur->rlink;
 }
 if(parent==NULL)
  return q;
 if(cur==parent->llink)
  parent->llink=q;
 else
  parent->rlink=q;
 freenode(cur);
 return root;
 }

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;
            }
          }
 }
```

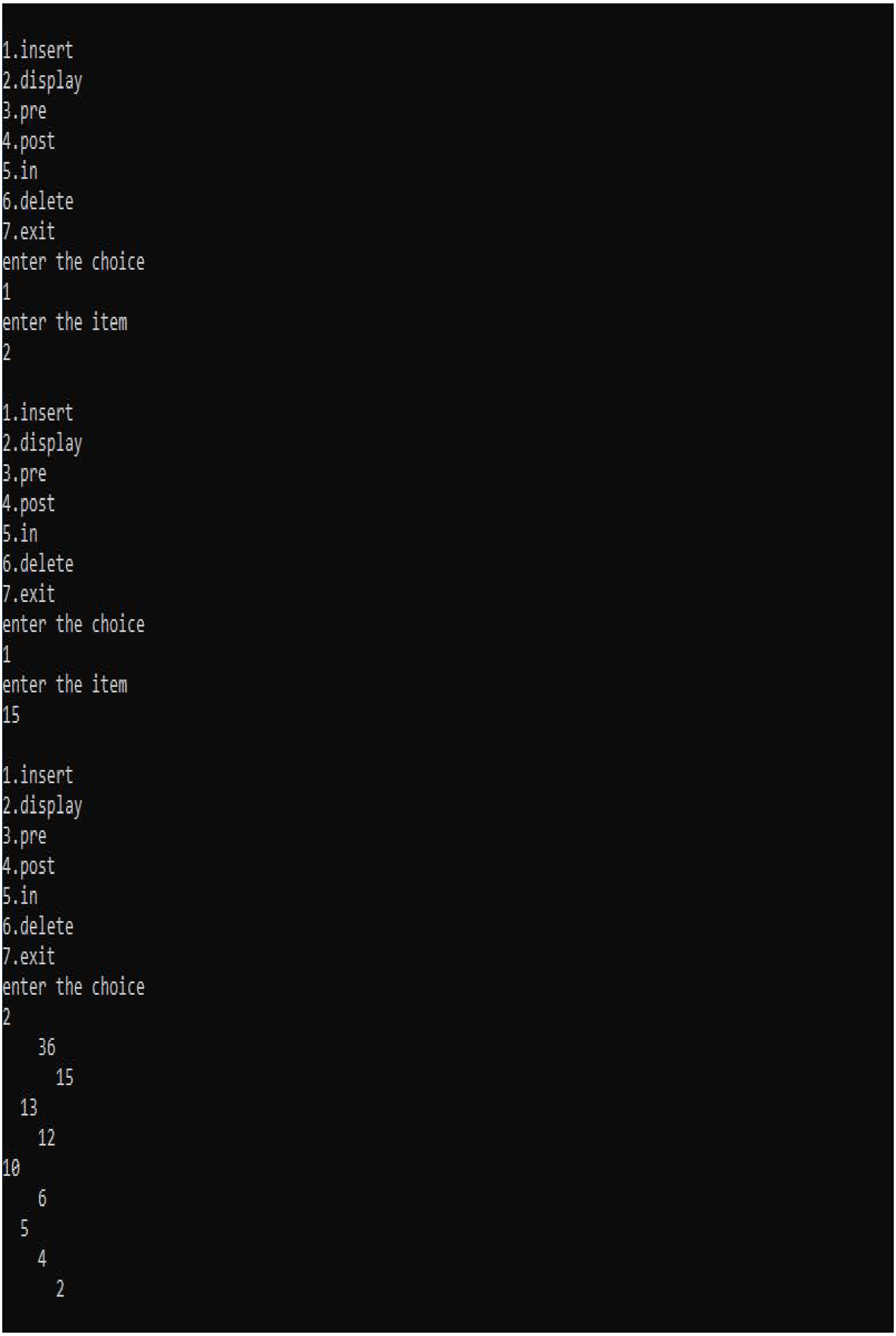
OUTPUT :



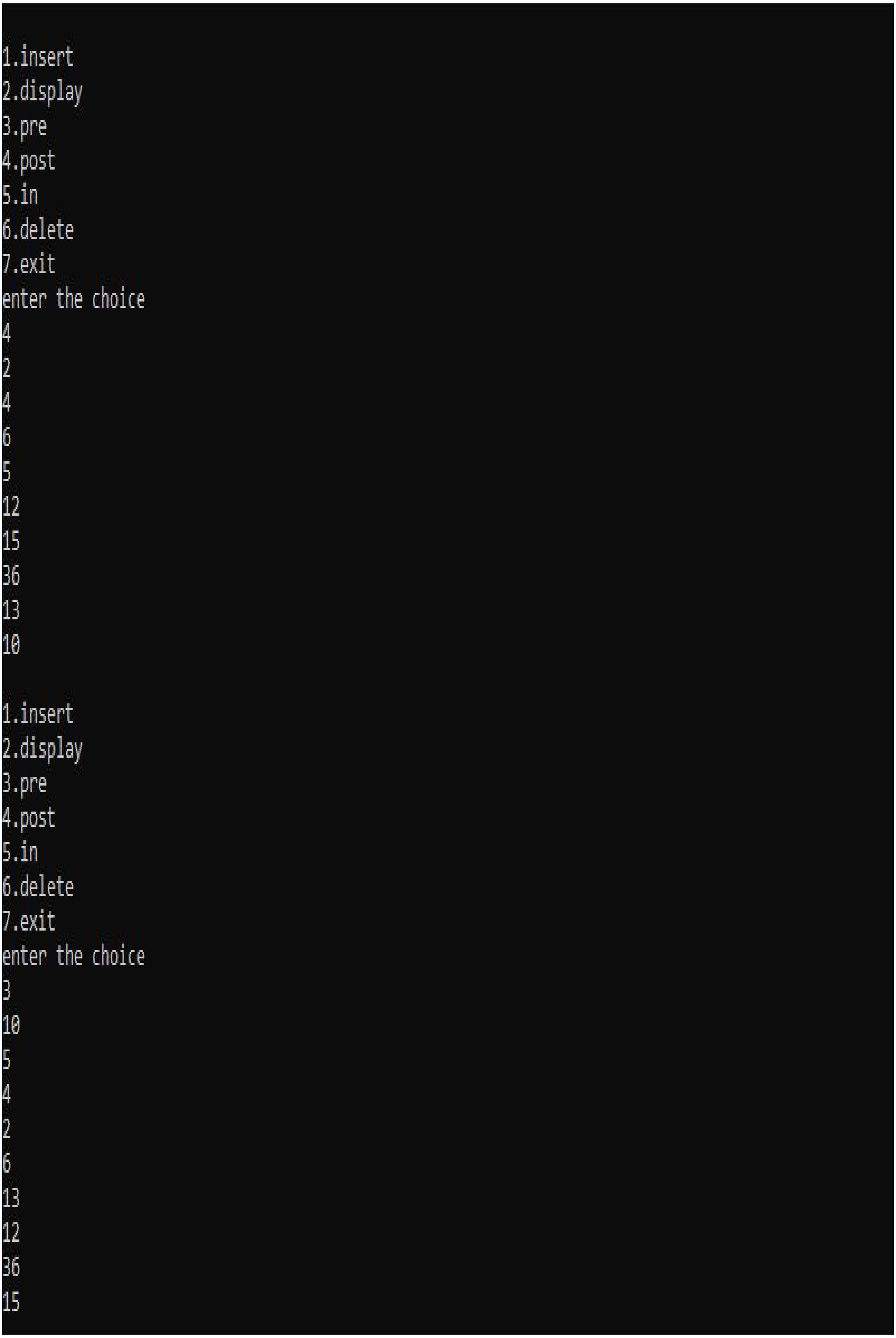
```

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
1
enter the item
5

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
13

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



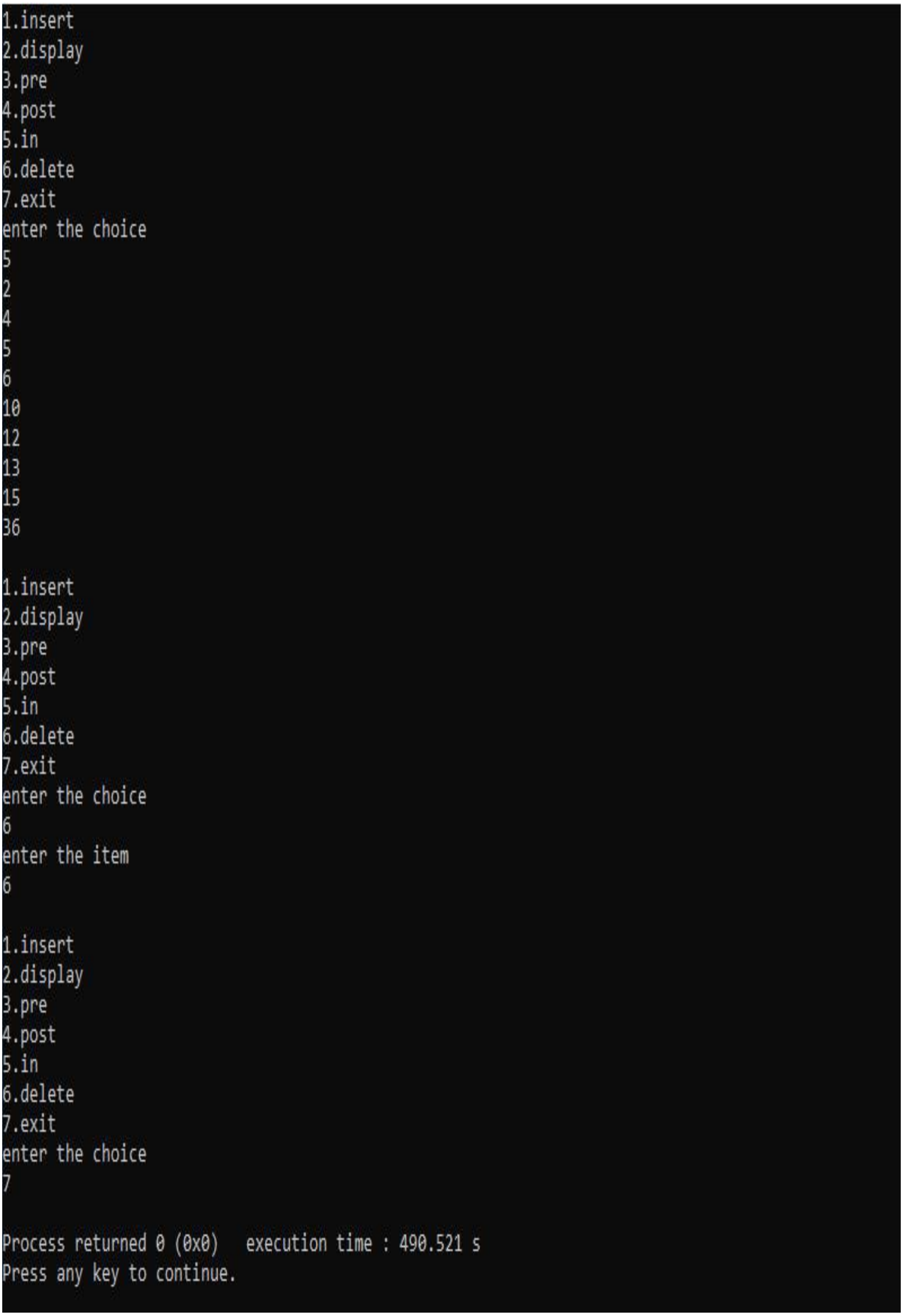
```
enter the choice
2
  13
10
  5

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
12

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
36

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
    36
  13
    12
10
  5
```



```

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

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
15

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
2
    36
      15
  13
    12
10
    6
  5
    4
      2
```



```

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
2
4
6
5
12
15
36
13
10

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
3
10
5
4
2
6
13
12
36
15
```



```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
2
4
5
6
10
12
13
15
36

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

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

Process returned 0 (0x0)   execution time : 490.521 s
Press any key to continue.
```