DS LAB-RECORD

NAME: N.Akhilesh Kumar Dutt

USN: 1BM19CS092

SECTION: 3B

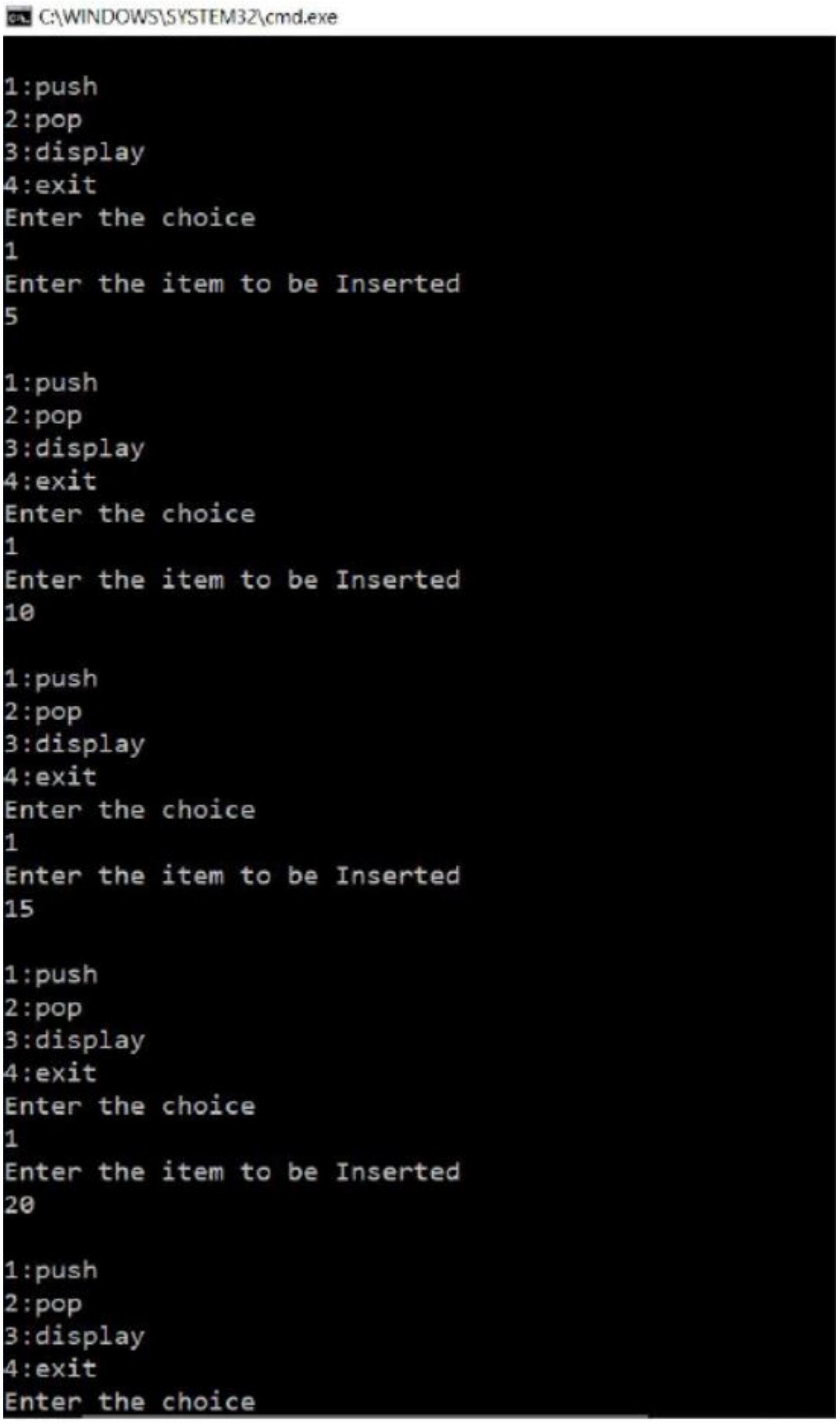
BATCH: 2

LAB PROGRAM 1 :

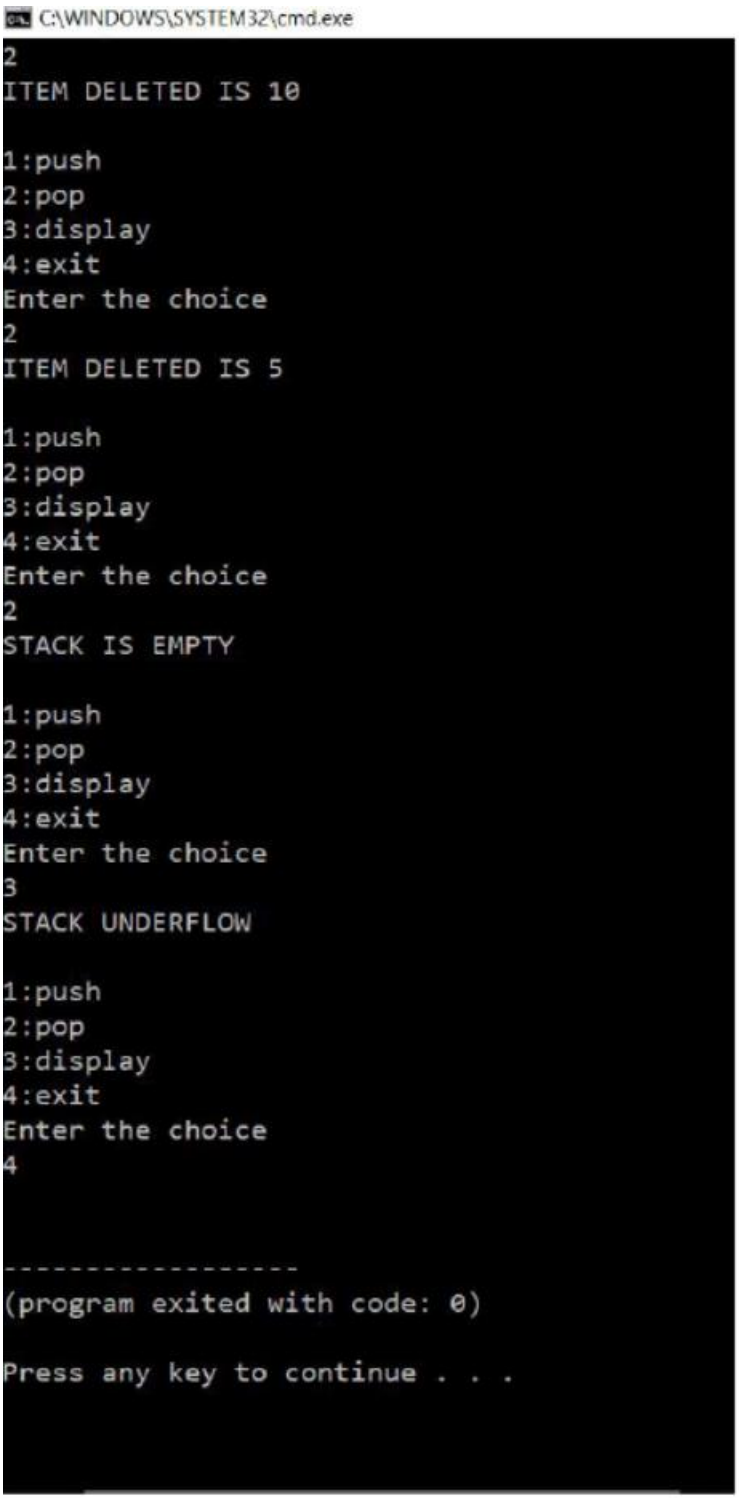
Write a program to simulate the working of stack using an array with the following : a) Push b) Pop c) Display The program should print appropriate messages for stack overflow, stack underflow.

```
#include<stdio.h>
#include<stdlib.h>
#define STACK_SIZE 5
int item;
int stack[10];
int top=-1;

void push()
{
  if(top==STACK_SIZE-1)
  {
    printf("STACK OVERFLOW\n");
    return ;
  }

  top=top+1;
  stack[top]=item;
}
int pop()
{
 if(top==-1)return -1;
 return stack[top--];
}
void display()
{
  int i;
  if(top==-1)
  {
    printf("STACK UNDERFLOW\n");
    return ;
  }

  printf("\nDISPLAYING CONTENTS OF STACK\n");

  for(i=0;i<=top;i++)
  {
              printf("%d\n",stack[i]);
  }
}
   int main()
{
              int Deleted_item;
              int choice;
              for(;;)
   {
               printf("\n1:push\n2:pop\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);
```

```
                push();
                break;

          case 2:Deleted_item=pop();
                if(Deleted_item==-1)
                {
                    printf("STACK IS EMPTY\n");
                 }
               else
            {
               printf("ITEM DELETED IS %d\n",Deleted_item);
            }
                     break;

          case 3:display();
                      break;
             default:exit(0);
                   return 0;
          }
        }
}
```


C:\WINDOWS\SYSTEM32\cmd.exe

```
1:push
2:pop
3:display
4:exit
Enter the choice
1
Enter the item to be Inserted
5

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
15

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
```



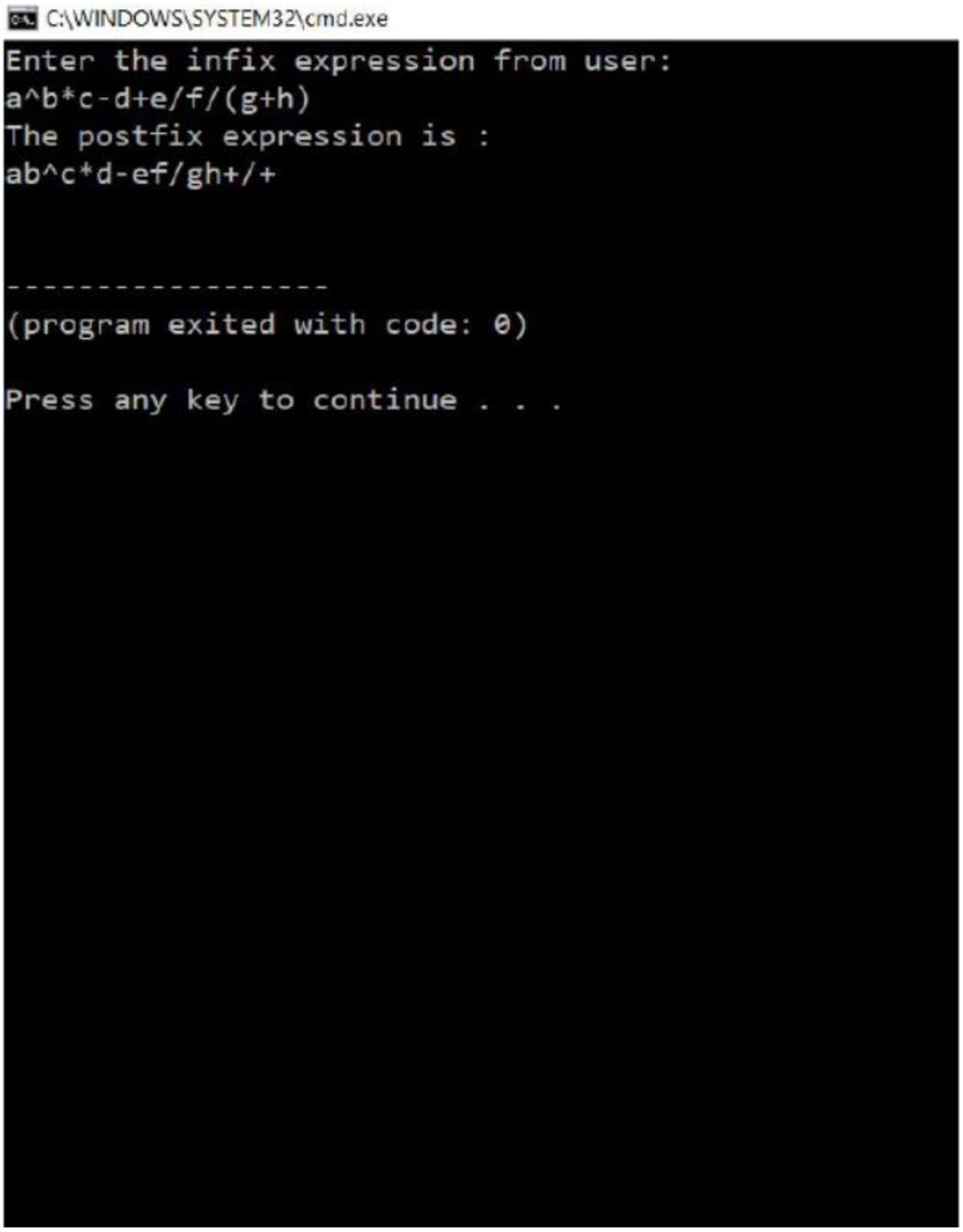
```
C:\WINDOWS\SYSTEM32\cmd.exe
2
ITEM DELETED IS 10

1:push
2:pop
3:display
4:exit
Enter the choice
2
ITEM DELETED IS 5

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 UNDERFLOW

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



-----------------
(program exited with code: 0)

Press any key to continue . . .
```

LAB PROGRAM 2 :

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

```
#include<stdio.h>
#include<string.h>
int 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,j,i;
        char Stack[30],symbol;
        top=-1;
        Stack[++top]='#';
        j=0;
        for(i=0;i<strlen(infix);i++)
        {
          symbol=infix[i];
          while(F(Stack[top])>G(symbol))
          {
            postfix[j]=Stack[top--];
            j++;
          }
          if(F(Stack[top])!=G(symbol))
          Stack[++top]=symbol;
          else
          top--;
        }
        while(Stack[top]!='#')
        {
            postfix[j++]=Stack[top--];
        }
        postfix[j]='\0';
}
int main()
{
    char infix[20];
    char postfix[20];
    printf("Enter the infix expression from user: \n");
    scanf("%s",infix);
    infix_postfix(infix,postfix);
    printf("The postfix expression is : \n");
    printf("%s\n",postfix);
    return 0;
}
```


C:\WINDOWS\SYSTEM32\cmd.exe

```
Enter the infix expression from user:
a^b*c-d+e/f/(g+h)
The postfix expression is :
ab^c*d-ef/gh+/+


------------------
(program exited with code: 0)

Press any key to continue . . .
```

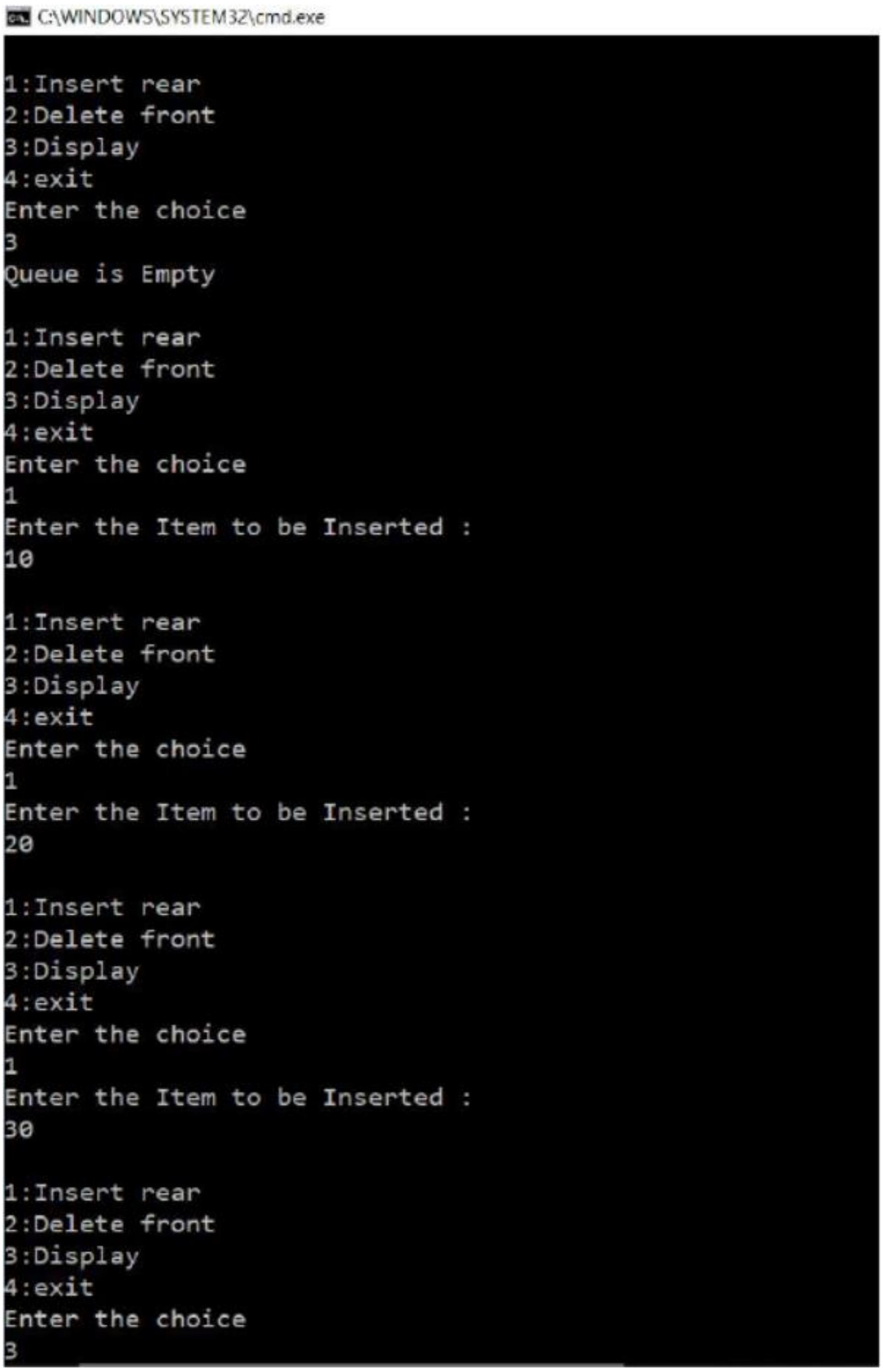
LAB PROGRAM 3 :

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 overflow conditions.

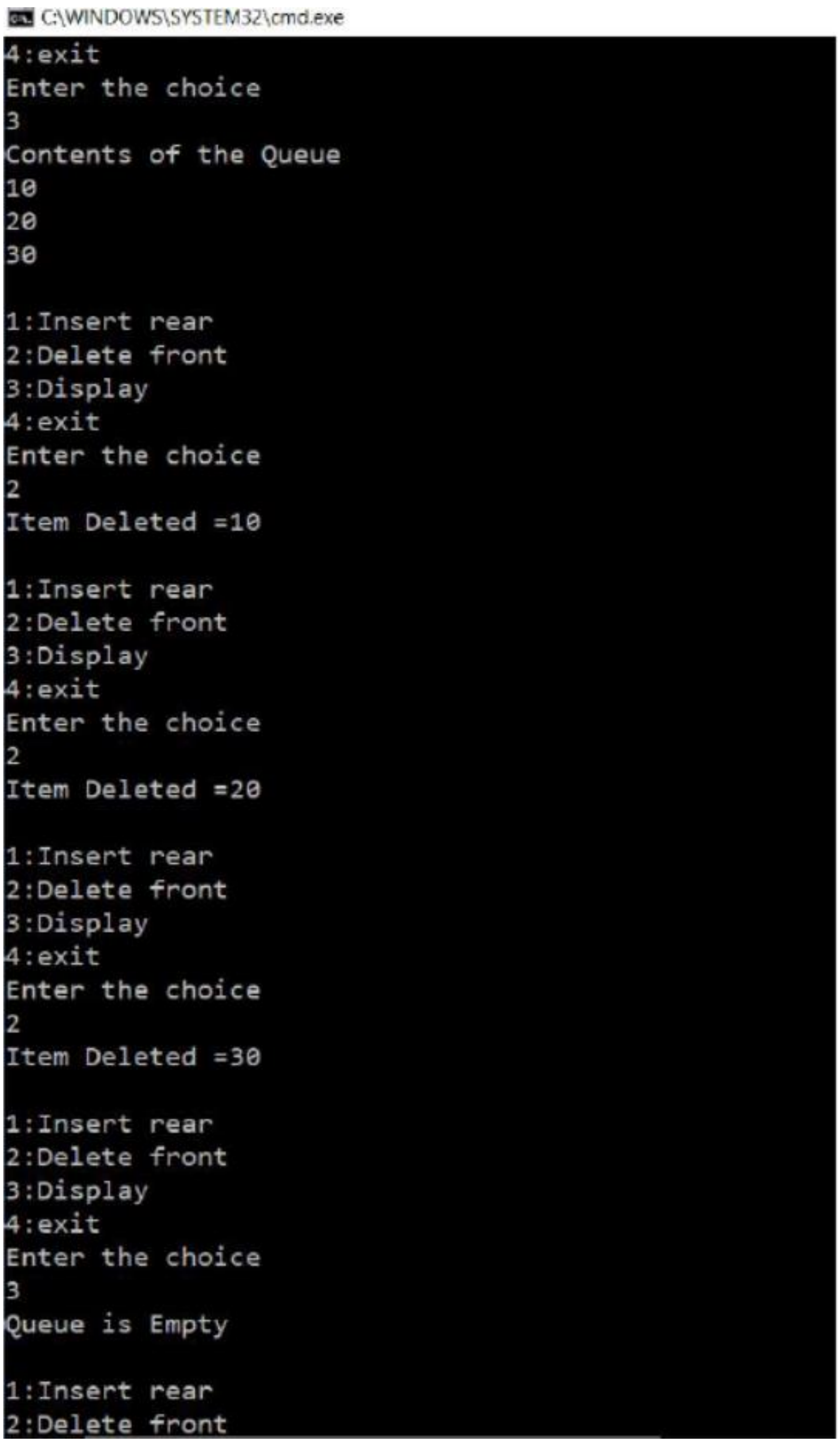
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define QUE_SIZE 3
int item,q[10];
int front=0,rear=-1;
void insertrear()
{
 if(rear==QUE_SIZE-1)
 {
  printf("Queue Overflow\n");
  return ;
 }
 rear=rear+1;
 q[rear]=item;
}
 int deletefront()
 {
  if(front>rear)
  {
   front=0;
   rear=-1;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

```
void displayQ()
{
 int i;
 if(front>rear)
 {
            printf("Queue is Empty\n");
            return;
 }
  printf("Contents of the Queue\n");
  for(i=front;i<=rear;i++)
  {
             printf("%d\n",q[i]);
            }
}
int main()
{
 int choice;
 for(;;)
 {
            printf("\n1:Insert rear\n2:Delete front\n3:Display\n4:exit\n");
            printf("Enter the choice\n");
            scanf("%d",&choice);
            switch(choice)
            {
   case 1: printf("Enter the Item to be Inserted :\n");
           scanf("%d",&item);
           insertrear();
           break;
   case 2: item=deletefront();
           if(item==1)
           printf("Queue is Empty\n");
```

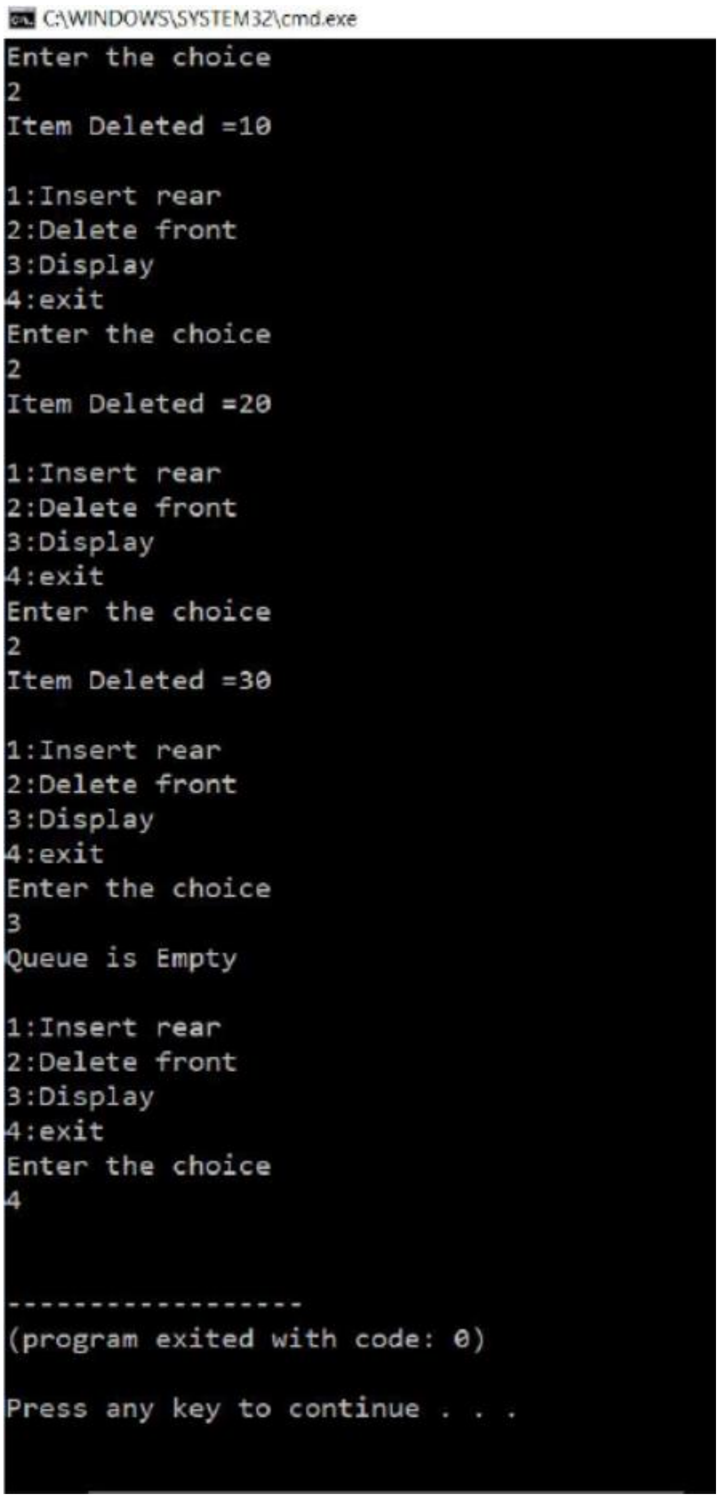
```
            else
            printf("Item Deleted =%d\n",item);
            break;
    case 3: displayQ();
            break;
    default : exit(0);
  }
 }
    return 0;
}
```


C:\WINDOWS\SYSTEM32\cmd.exe

```
1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
3
Queue is Empty

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
1
Enter the Item to be Inserted :
10

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
1
Enter the Item to be Inserted :
20

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
1
Enter the Item to be Inserted :
30

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
3
```


C:\WINDOWS\SYSTEM32\cmd.exe

```
4:exit
Enter the choice
3
Contents of the Queue
10
20
30

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
2
Item Deleted =10

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
2
Item Deleted =20

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
2
Item Deleted =30

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
3
Queue is Empty

1:Insert rear
2:Delete front
```


C:\WINDOWS\SYSTEM32\cmd.exe

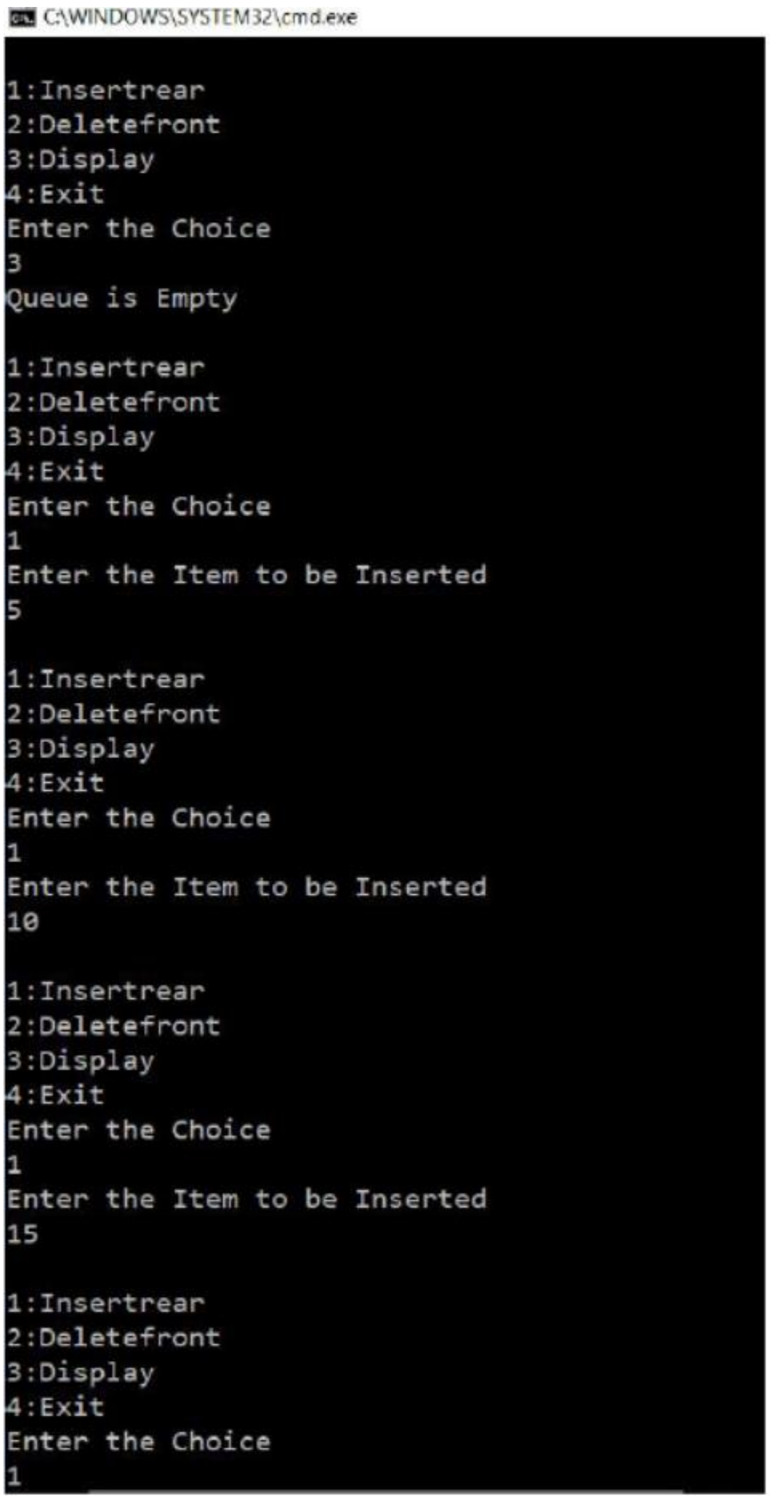
```
Enter the choice
2
Item Deleted =10

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
2
Item Deleted =20

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
2
Item Deleted =30

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
3
Queue is Empty

1:Insert rear
2:Delete front
3:Display
4:exit
Enter the choice
4



------------------
(program exited with code: 0)

Press any key to continue . . .
```

LAB PROGRAM 4 :

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

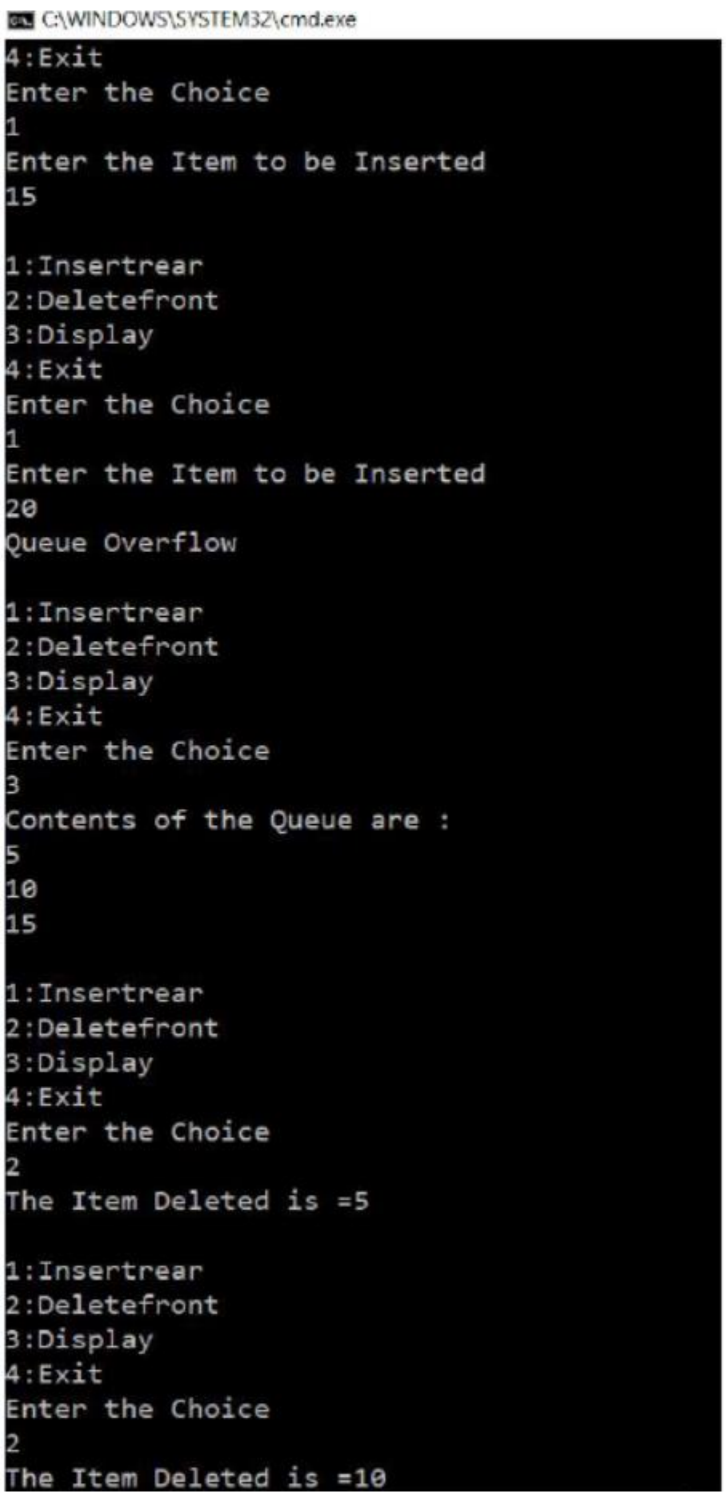
```
#include<stdio.h>
#include<process.h>
#define QUE_SIZE 3
int item,front=0,rear=-1,q[QUE_SIZE],count=0;
void insertrear()
{
if(count==QUE_SIZE)
{
printf("Queue Overflow\n");
return;
}
rear=(rear+1)%QUE_SIZE;
q[rear]=item;
count++;
}
int deletefront()
{
if(count==0) return -1;
item=q[front];
front=(front+1)%QUE_SIZE;
count=count-1;
return item;
}
void displayQ()
{
int i,f;
if(count==0)
{
printf("Queue is Empty\n");
return;
}
```

```
f=front;

printf("Contents of the Queue are :\n");

for(i=1;i<=count;i++)

{

printf("%d\n",q[f]);

f=(f+1)%QUE_SIZE;

}

}

int 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("The Item Deleted is =%d\n",item);

             break;

 case 3:displayQ();

             break;
 default:exit(0);
}
}
 return 0;
}
```

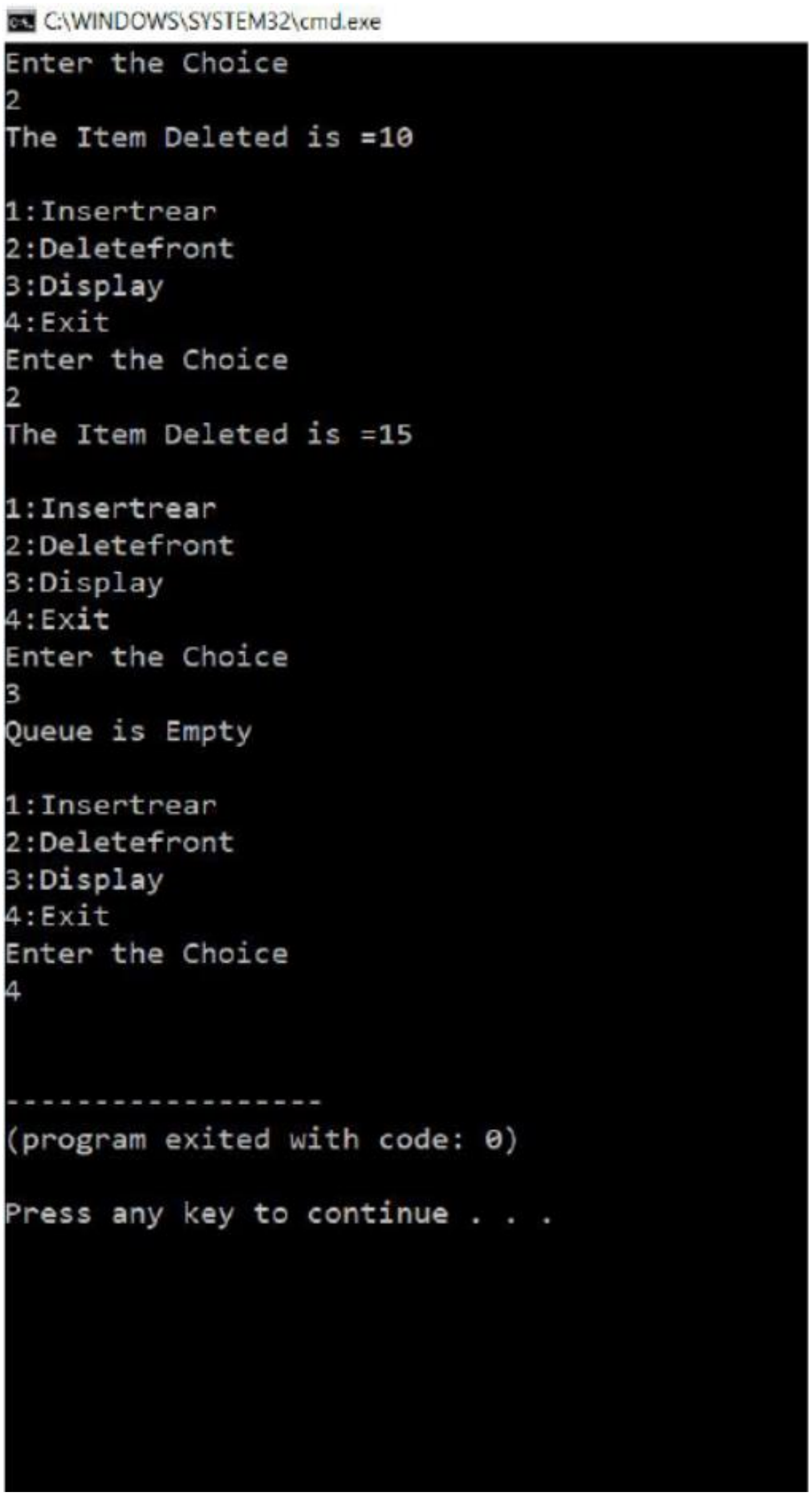


C:\WINDOWS\SYSTEM32\cmd.exe

```

1:Insertrear
2:Deletefront
3:Display
4:Exit
Enter the Choice
3
Queue is Empty

1:Insertrear
2:Deletefront
3:Display
4:Exit
Enter the Choice
1
Enter the Item to be Inserted
5

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
15

1:Insertrear
2:Deletefront
3:Display
4:Exit
Enter the Choice
1
```



```
C:\WINDOWS\SYSTEM32\cmd.exe
4:Exit
Enter the Choice
1
Enter the Item to be Inserted
15

1:Insertrear
2:Deletefront
3:Display
4:Exit
Enter the Choice
1
Enter the Item to be Inserted
20
Queue Overflow

1:Insertrear
2:Deletefront
3:Display
4:Exit
Enter the Choice
3
Contents of the Queue are :
5
10
15

1:Insertrear
2:Deletefront
3:Display
4:Exit
Enter the Choice
2
The Item Deleted is =5

1:Insertrear
2:Deletefront
3:Display
4:Exit
Enter the Choice
2
The Item Deleted is =10
```



C:\WINDOWS\SYSTEM32\cmd.exe

```
Enter the Choice
2
The Item Deleted is =10

1:Insertrear
2:Deletefront
3:Display
4:Exit
Enter the Choice
2
The Item Deleted is =15

1:Insertrear
2:Deletefront
3:Display
4:Exit
Enter the Choice
3
Queue is Empty

1:Insertrear
2:Deletefront
3:Display
4:Exit
Enter the Choice
4



------------------
(program exited with code: 0)

Press any key to continue . . .
```

LAB PROGRAM 5 and 6:

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.

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.

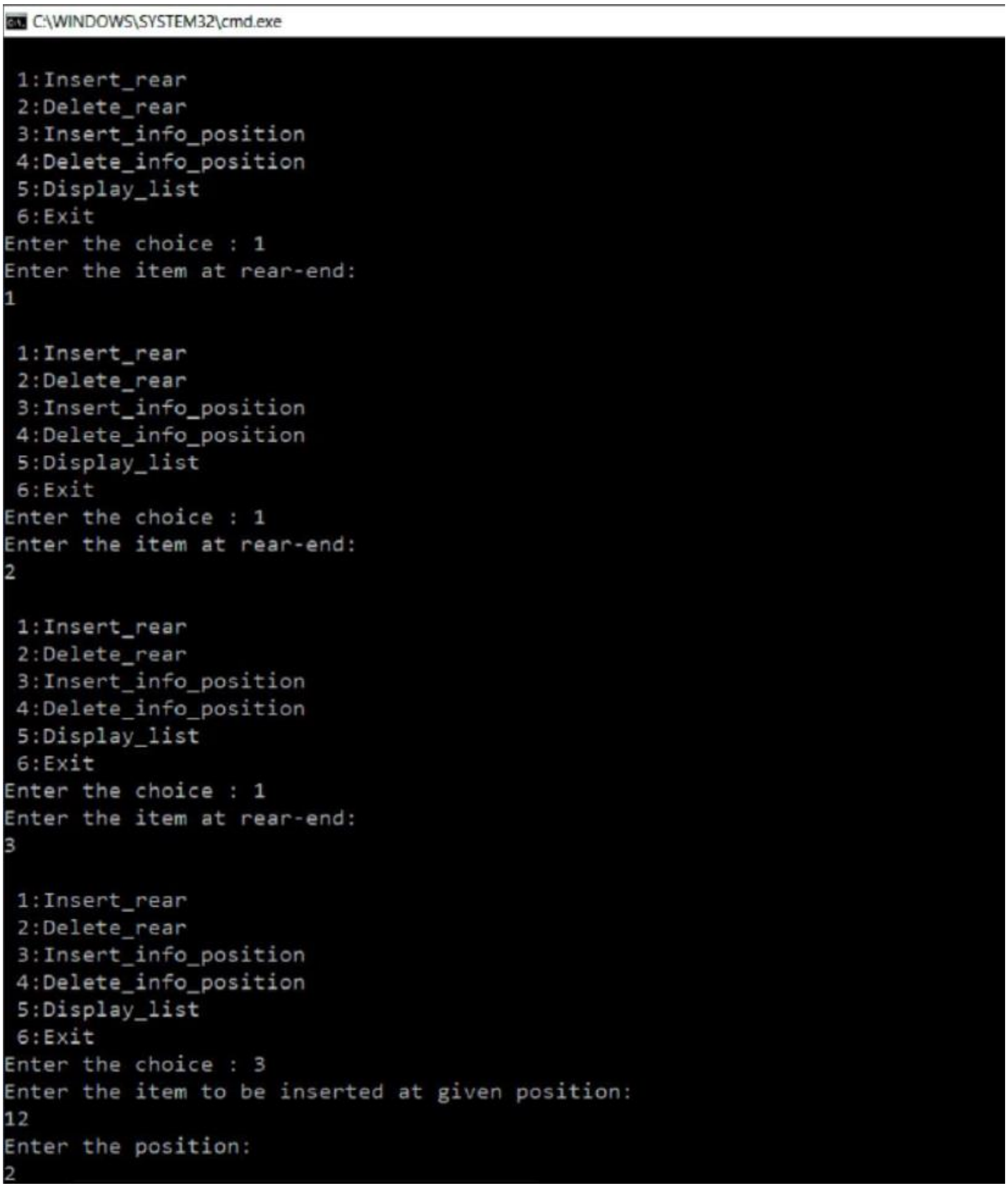
```
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
struct node
{
 int info;
 struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
 printf("Mem full\n");
 exit(0);
 }
 return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
```

```
 return temp;
cur=first;
while(cur->link!=NULL)
 cur=cur->link;
cur->link=temp;
return first;
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("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;
```

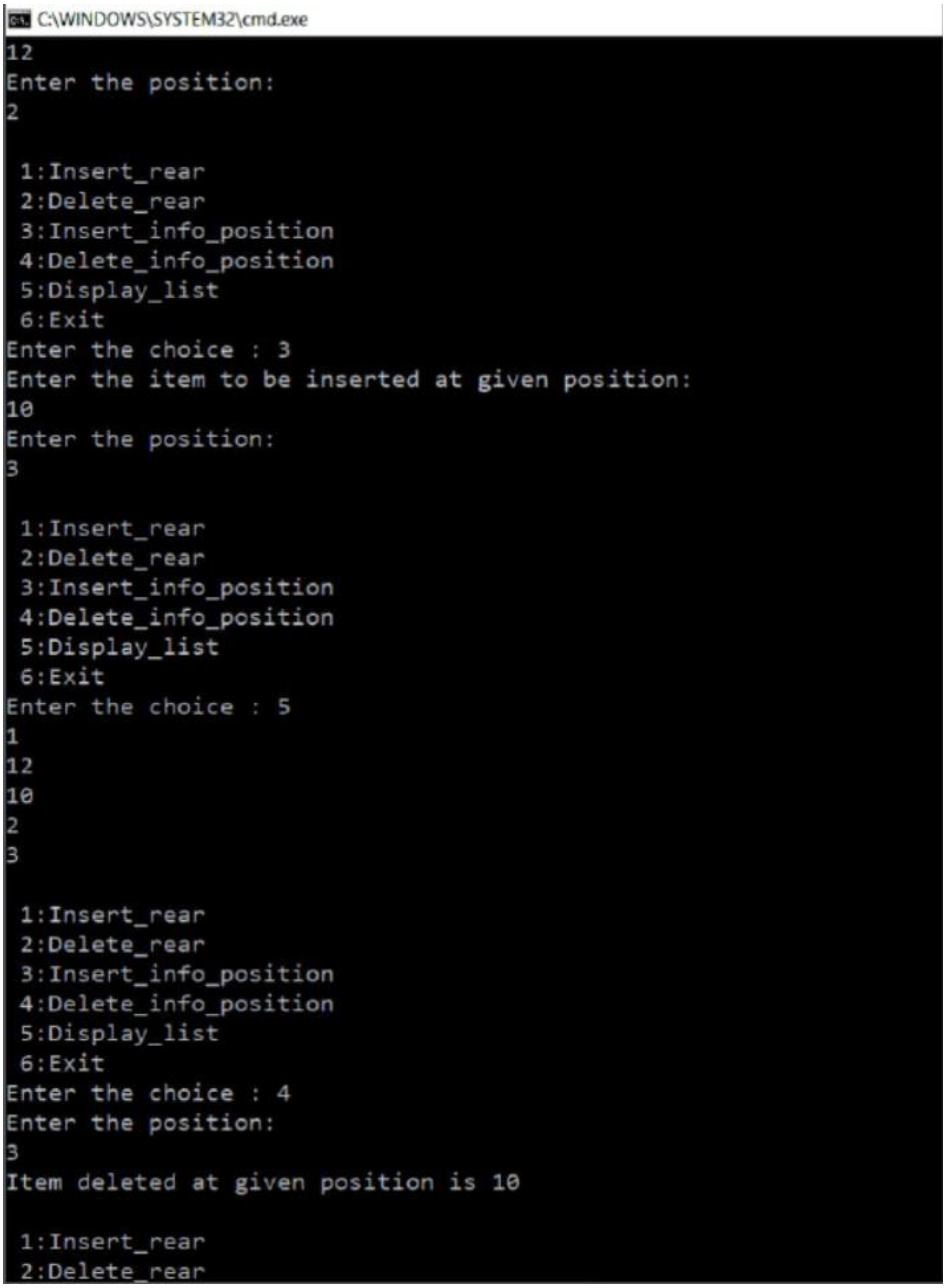
```
return first;

}

NODE 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 first;

}

if(pos==1)

{

temp->link=first;

first=temp;

return temp;

}

count=1;

prev=NULL;

cur=first;

while(cur!=NULL&&count!=pos)

{

prev=cur;

cur=cur->link;
count++;

}

if(count==pos)

{

prev->link=temp;

temp->link=cur;

return first;

}

printf("Invalid position\n");

return first;

}

NODE delete_pos(int pos,NODE first)

{

NODE cur;

NODE prev;

int count,flag=0;

if(first==NULL || pos<0)

{

printf("Invalid position\n");

return NULL;

}

if(pos==1)

{

cur=first;

first=first->link;

freenode(cur);

return first;

}

prev=NULL;

cur=first;

count=1;
```

```
while(cur!=NULL)
{
if(count==pos){flag=1;break;}
count++;
prev=cur;
cur=cur->link;
}
if(flag==0)
{
printf("Invalid position\n");
return first;
}
printf("Item deleted at given position is %d\n",cur->info);
prev->link=cur->link;
freenode(cur);
return first;
}
void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("List is empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
 {
 printf("%d\n",temp->info);
 }
}
int main()
{
int item,choice,pos;
NODE first=NULL;
```

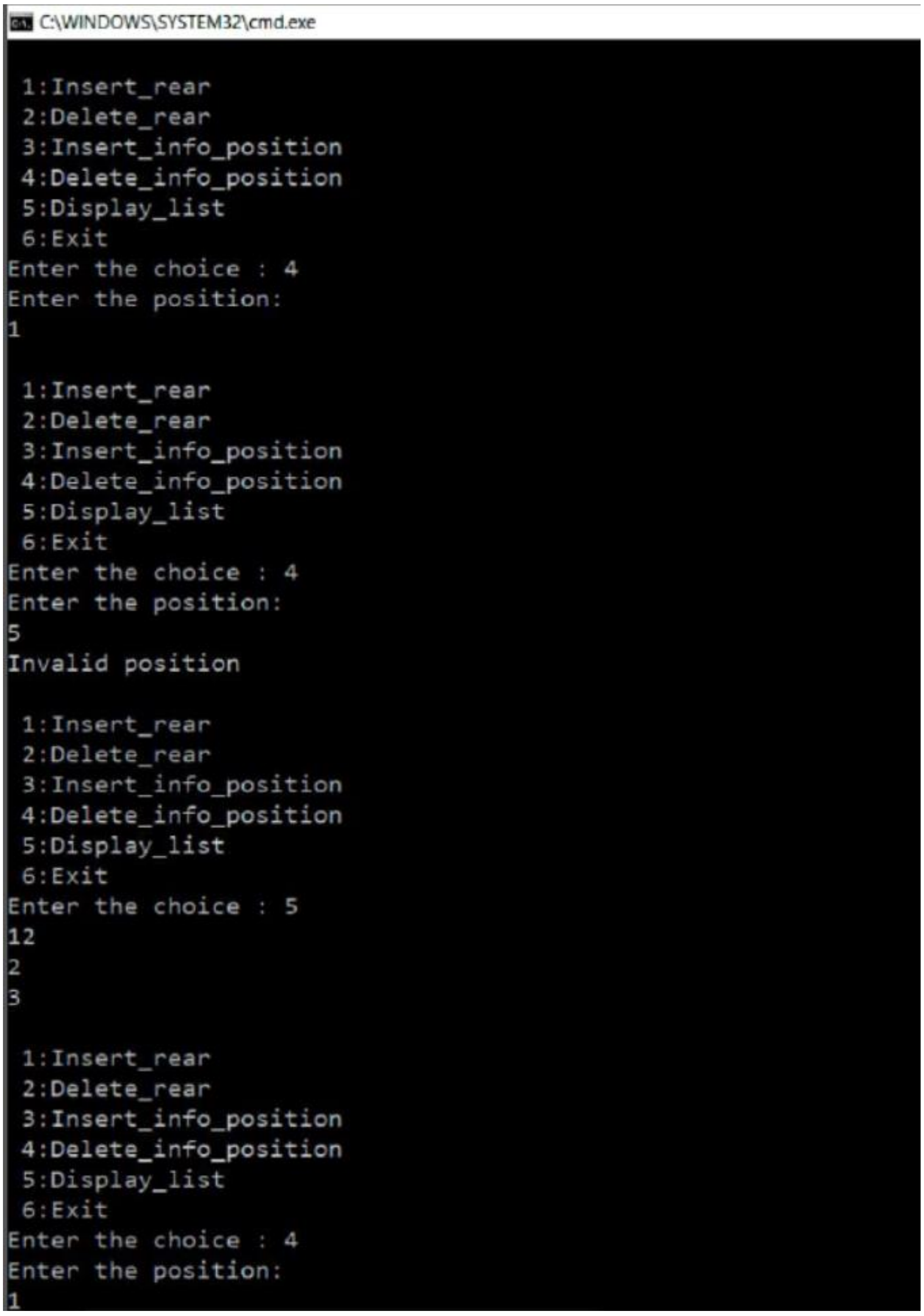
```
for(;;)
{
printf("\n 1:Insert_rear\n 2:Delete_rear\n");
printf(" 3:Insert_info_position\n 4:Delete_info_position\n 5:Display_list\n 6:Exit\n");
printf("Enter the choice:\n");
scanf("%d",&choice);
switch(choice)
 {
 case 1:printf("Enter the item at rear-end:\n");
 scanf("%d",&item);
 first=insert_rear(first,item);
 break;
 case 2:first=delete_rear(first);
 break;
 case 3:printf("Enter the item to be inserted at given position:\n");
 scanf("%d",&item);
 printf("Enter the position:\n");
 scanf("%d",&pos);
 first=insert_pos(item,pos,first);
 break;
 case 4:printf("Enter the position:\n");
 scanf("%d",&pos);
 first=delete_pos(pos,first);
 break;
 case 5:display(first);
 break;
 default:exit(0);
 break;
 }
}
return 0; }
```



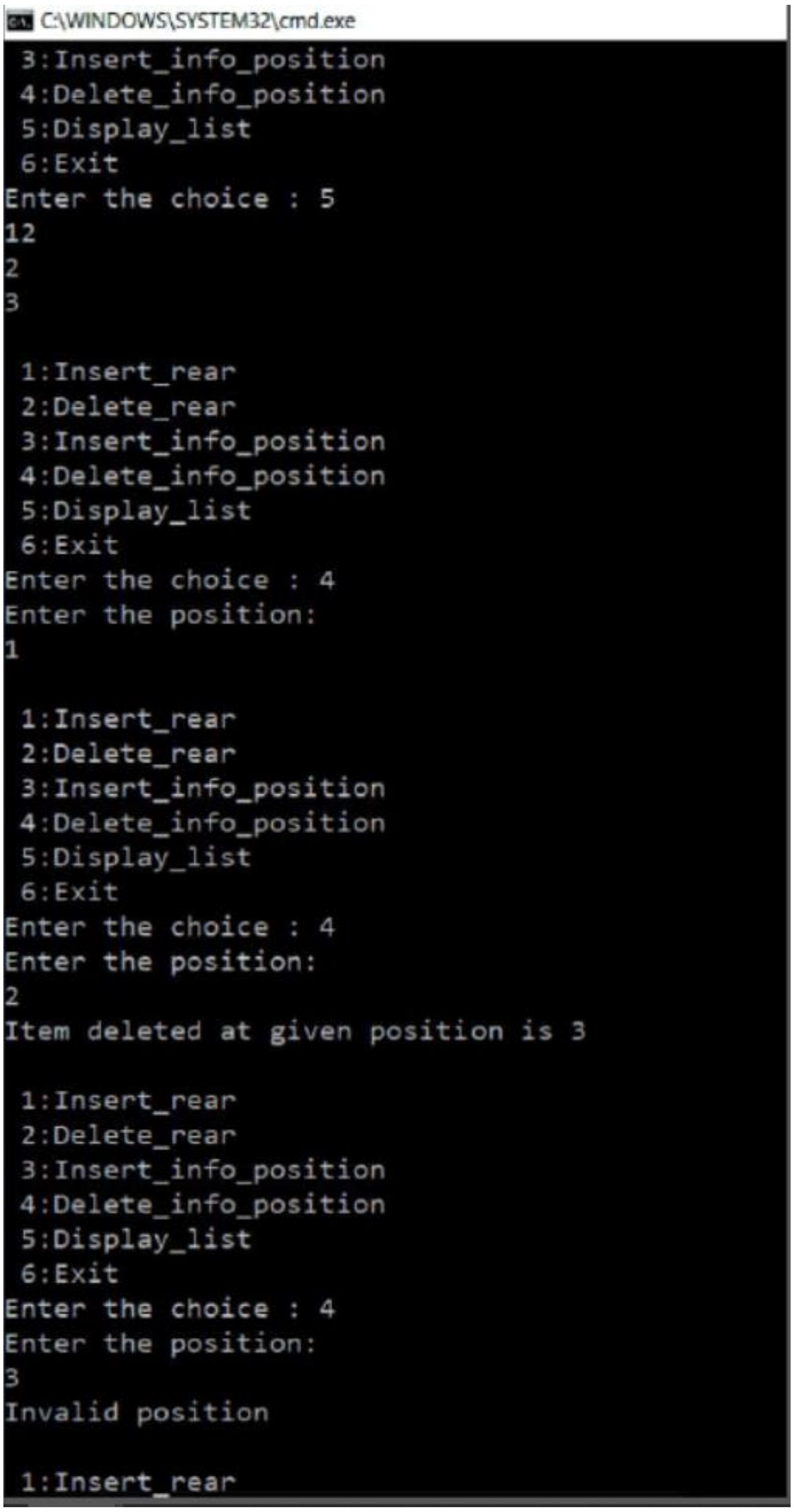
```
 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 1
Enter the item at rear-end:
1

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

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

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 3
Enter the item to be inserted at given position:
12
Enter the position:
2
```



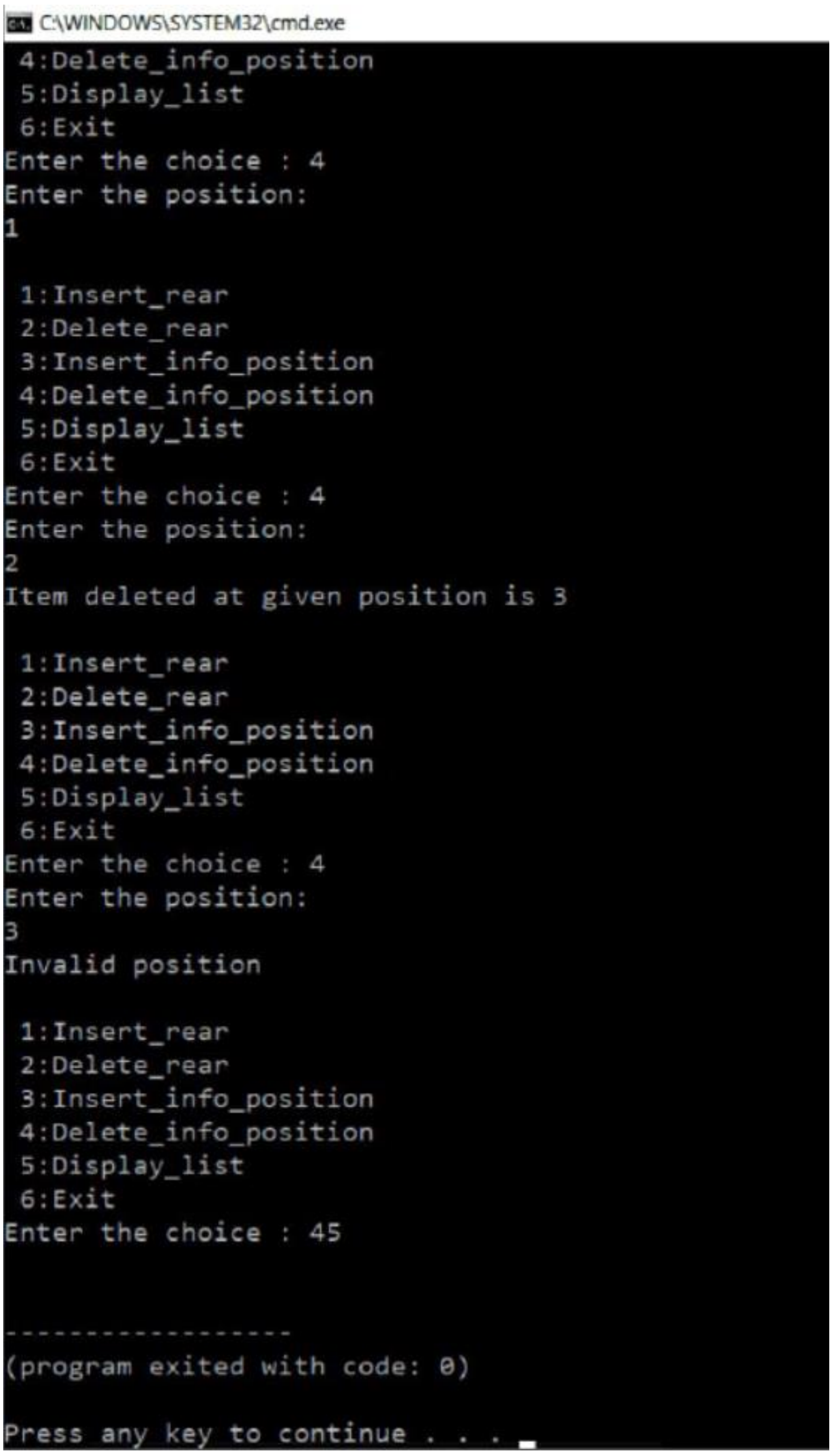
```
C:\WINDOWS\SYSTEM32\cmd.exe
12
Enter the position:
2

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 3
Enter the item to be inserted at given position:
10
Enter the position:
3

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 5
1
12
10
2
3

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 4
Enter the position:
3
Item deleted at given position is 10

 1:Insert_rear
 2:Delete_rear
```



```
 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 4
Enter the position:
1

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 4
Enter the position:
5
Invalid position

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 5
12
2
3

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 4
Enter the position:
1
```



```
C:\WINDOWS\SYSTEM32\cmd.exe
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 5
12
2
3

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 4
Enter the position:
1

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 4
Enter the position:
2
Item deleted at given position is 3

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 4
Enter the position:
3
Invalid position

 1:Insert_rear
```



```
C:\WINDOWS\SYSTEM32\cmd.exe
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 4
Enter the position:
1

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 4
Enter the position:
2
Item deleted at given position is 3

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 4
Enter the position:
3
Invalid position

 1:Insert_rear
 2:Delete_rear
 3:Insert_info_position
 4:Delete_info_position
 5:Display_list
 6:Exit
Enter the choice : 45


------------------
(program exited with code: 0)

Press any key to continue . . .
```

LAB PROGRAM 7 :

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

```
#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;
```

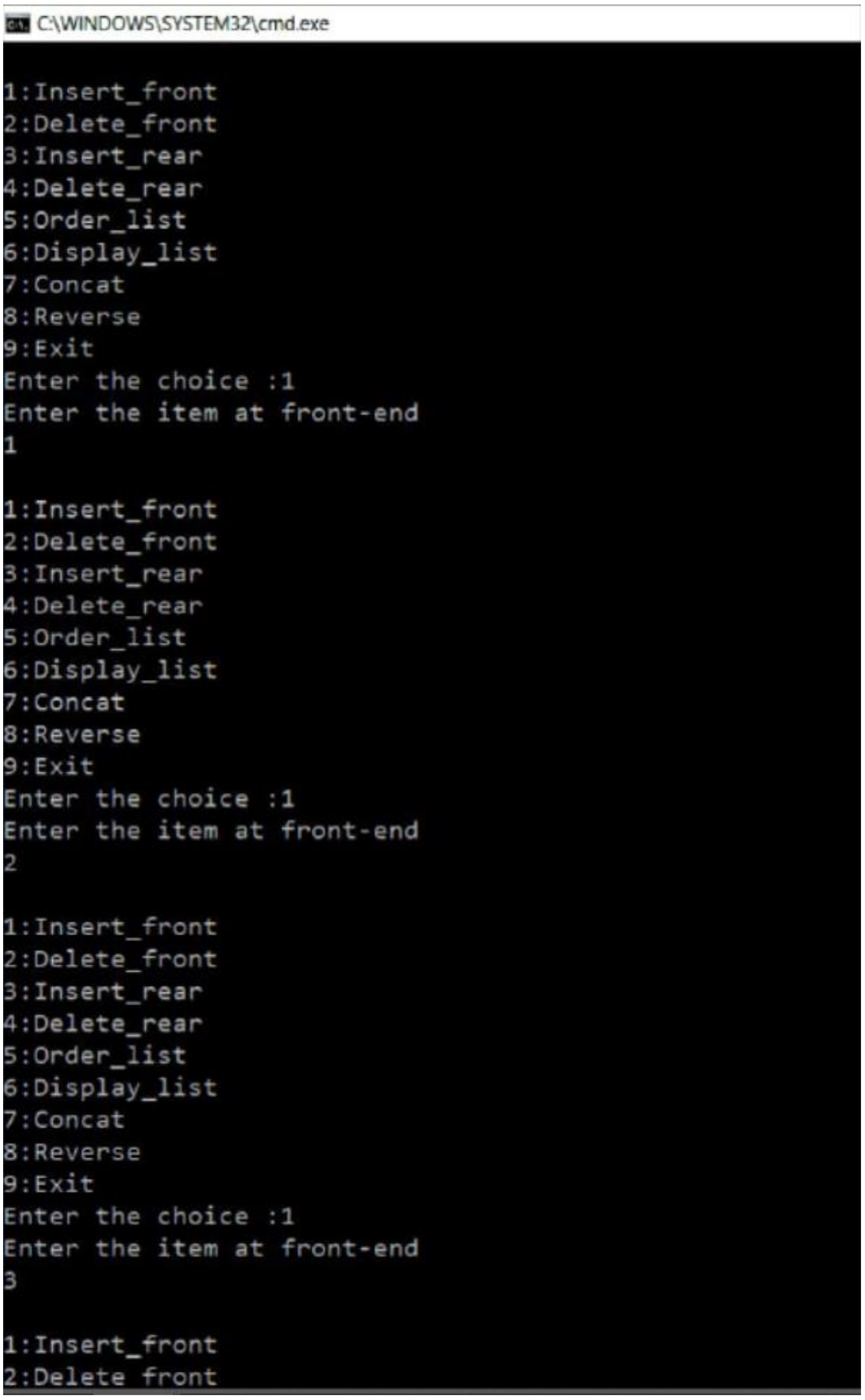
```
return first;
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("List is empty cannot delete\n");
return first;
}
if(first->link==NULL)
{
printf("Item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;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

void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("List empty cannot display items\n");
 printf("Contents of the list:\n");
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\n",temp->info);
  }
```

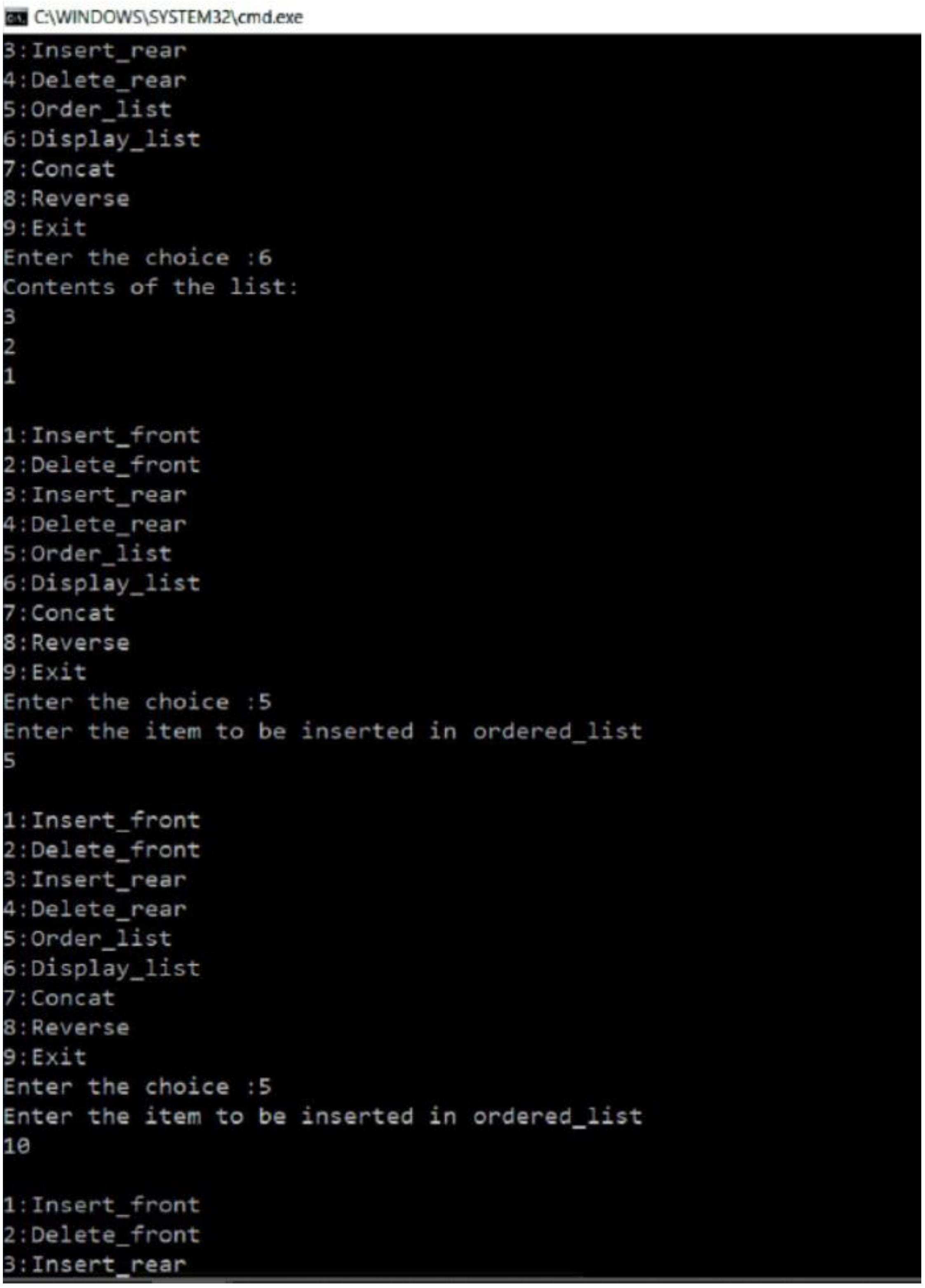
```
}
NODE concat(NODE first,NODE second)
{
 NODE cur;
 if(first==NULL)
  return second;
 if(second==NULL)
  return first;
 cur=first;
 while(cur->link!=NULL)
  cur=cur->link;
 cur->link=second;
 return first;
}
NODE reverse(NODE first)
 {
 NODE cur,temp;
 cur=NULL;
 while(first!=NULL)
  {
   temp=first;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

int main()
{
int item,choice,n,i;
```

```
NODE first=NULL,a,b;
for(;;)
{
printf("\n1:Insert_front\n2:Delete_front\n3:Insert_rear\n4:Delete_rear\n");
printf("5:Order_list\n6:Display_list\n7:Concat\n8:Reverse\n9:Exit\n");
printf("Enter the choice :");
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:display(first);
              break;
  case 7:printf("Enter the no of nodes in 1\n");
                          scanf("%d",&n);
                          a=NULL;
```

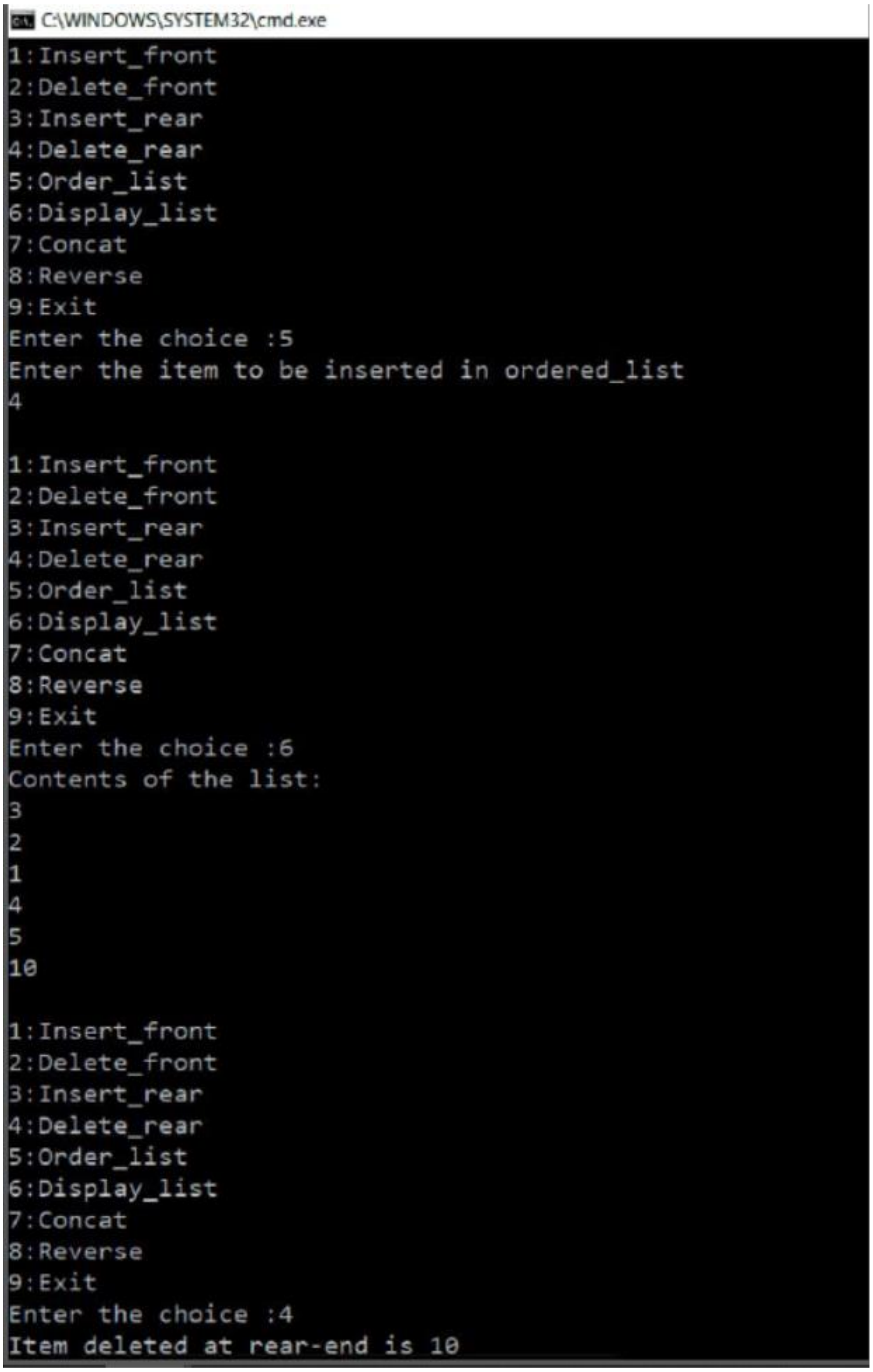
```
                        for(i=0;i<n;i++)
                         {
                          printf("Enter the item\n");
                          scanf("%d",&item);
                          a=insert_rear(a,item);
                         }
                         printf("Enter the no of nodes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



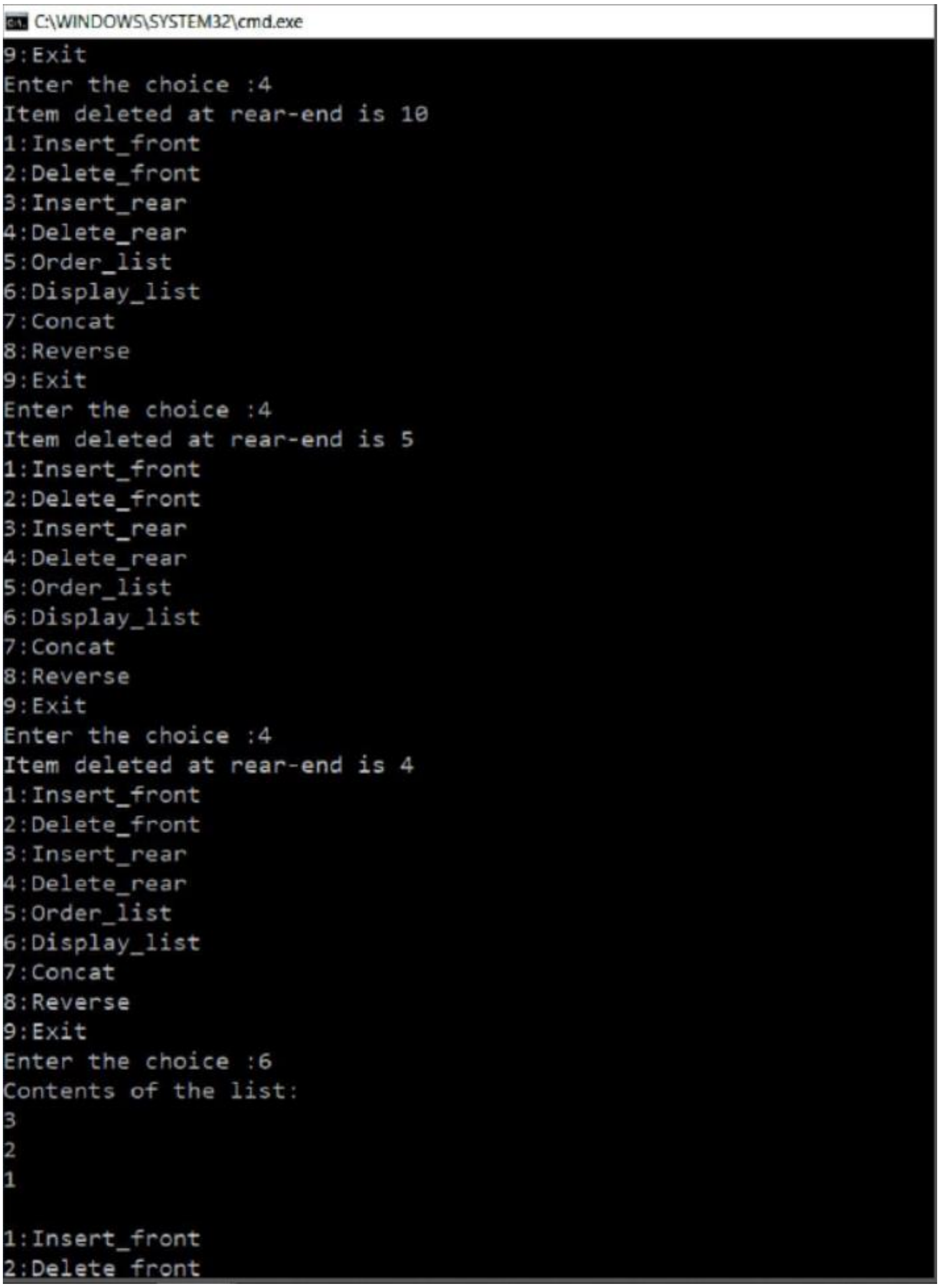
```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :1
Enter the item at front-end
1

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :1
Enter the item at front-end
2

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :1
Enter the item at front-end
3

1:Insert_front
2:Delete_front
```


C:\WINDOWS\SYSTEM32\cmd.exe

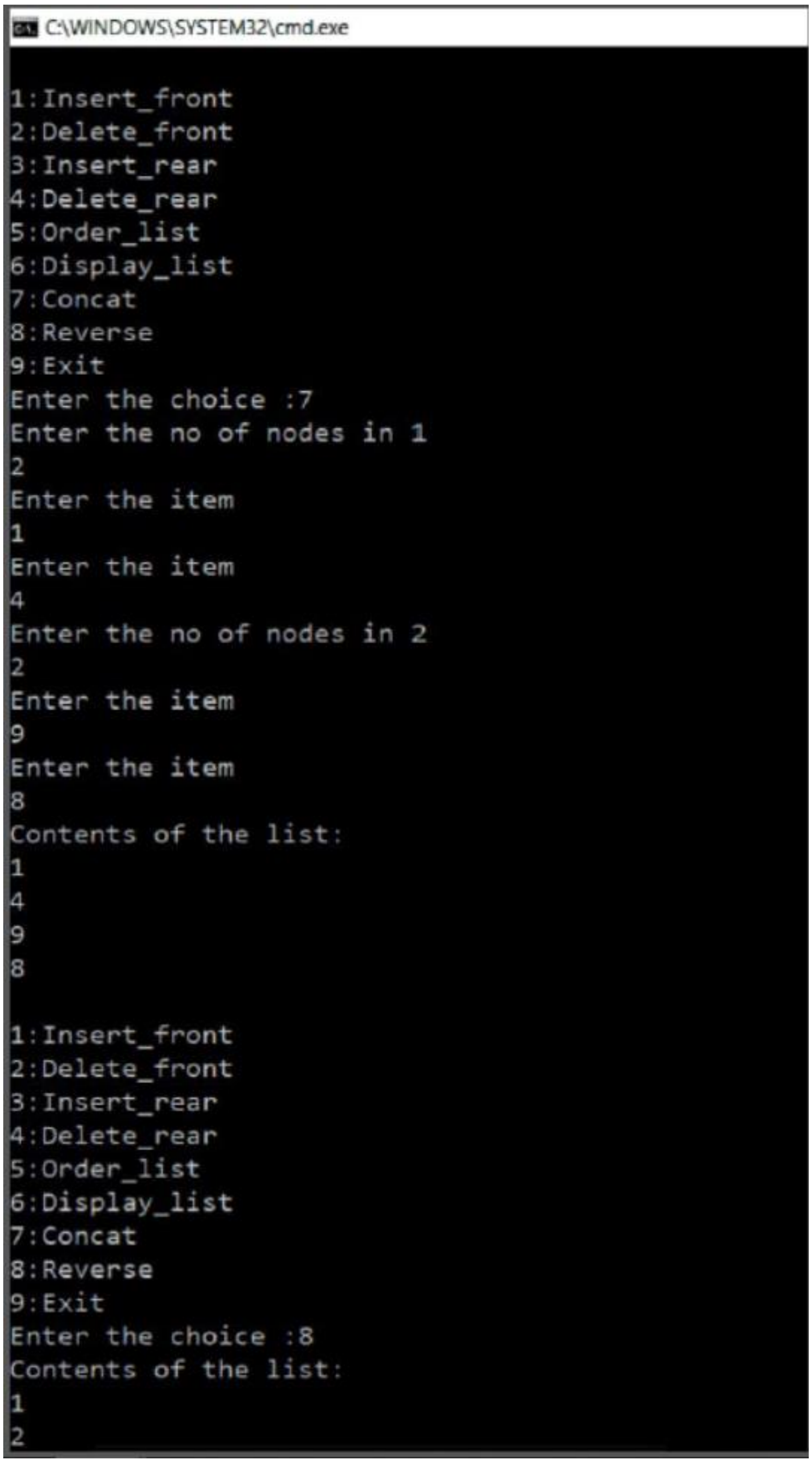
```
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :6
Contents of the list:
3
2
1

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :5
Enter the item to be inserted in ordered_list
5

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :5
Enter the item to be inserted in ordered_list
10

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


C:\WINDOWS\SYSTEM32\cmd.exe

```
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :5
Enter the item to be inserted in ordered_list
4

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :6
Contents of the list:
3
2
1
4
5
10

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :4
Item deleted at rear-end is 10
```

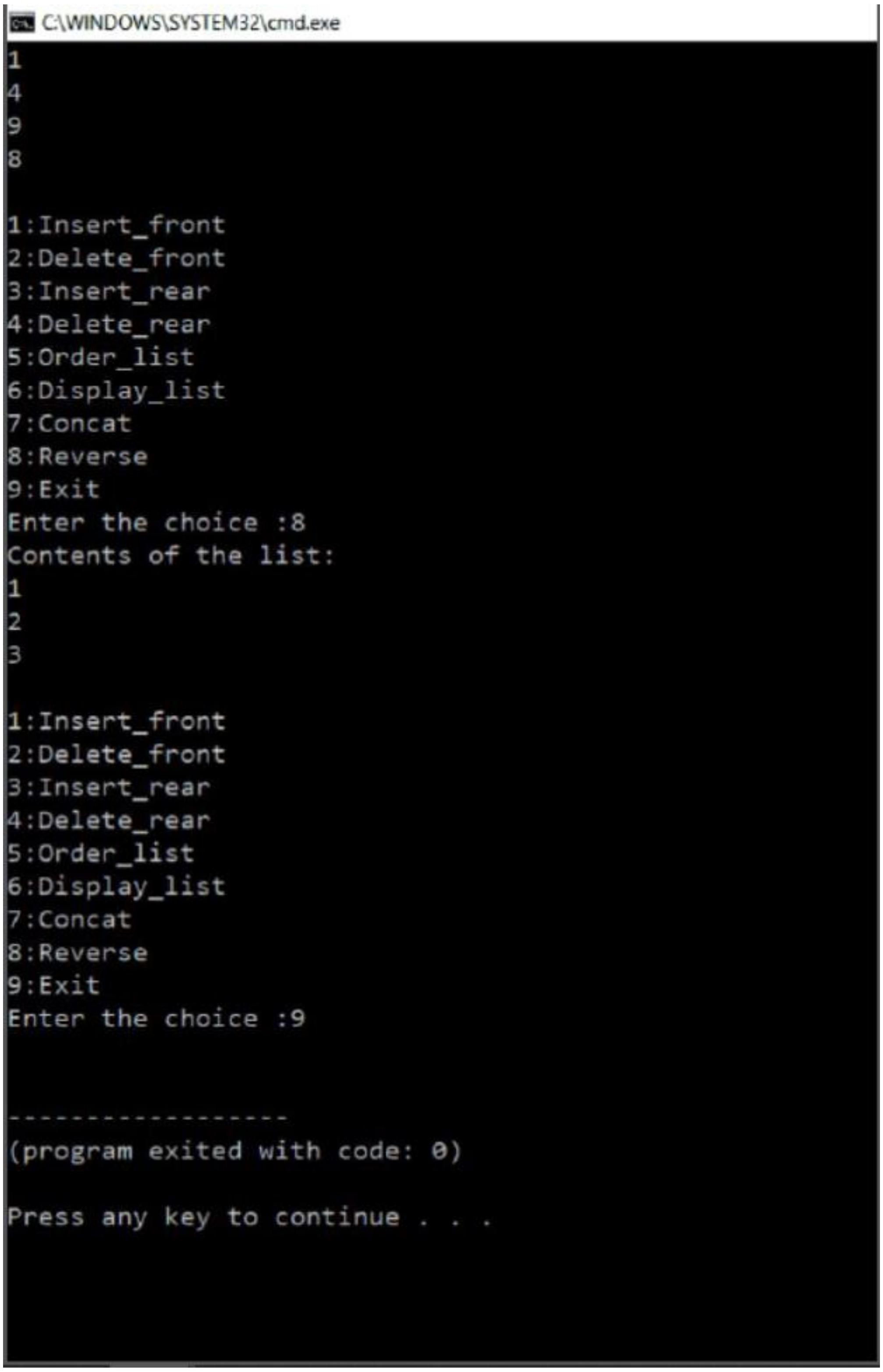


```
9:Exit
Enter the choice :4
Item deleted at rear-end is 10
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :4
Item deleted at rear-end is 5
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :4
Item deleted at rear-end is 4
1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :6
Contents of the list:
3
2
1

1:Insert_front
2:Delete_front
```


C:\WINDOWS\SYSTEM32\cmd.exe

```

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :7
Enter the no of nodes in 1
2
Enter the item
1
Enter the item
4
Enter the no of nodes in 2
2
Enter the item
9
Enter the item
8
Contents of the list:
1
4
9
8

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :8
Contents of the list:
1
2
```

LAB PROGRAM 8 :



```
C:\WINDOWS\SYSTEM32\cmd.exe
1
4
9
8

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :8
Contents of the list:
1
2
3

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5:Order_list
6:Display_list
7:Concat
8:Reverse
9:Exit
Enter the choice :9


------------------
(program exited with code: 0)

Press any key to continue . . .
```

LAB PROGRAM 8 :

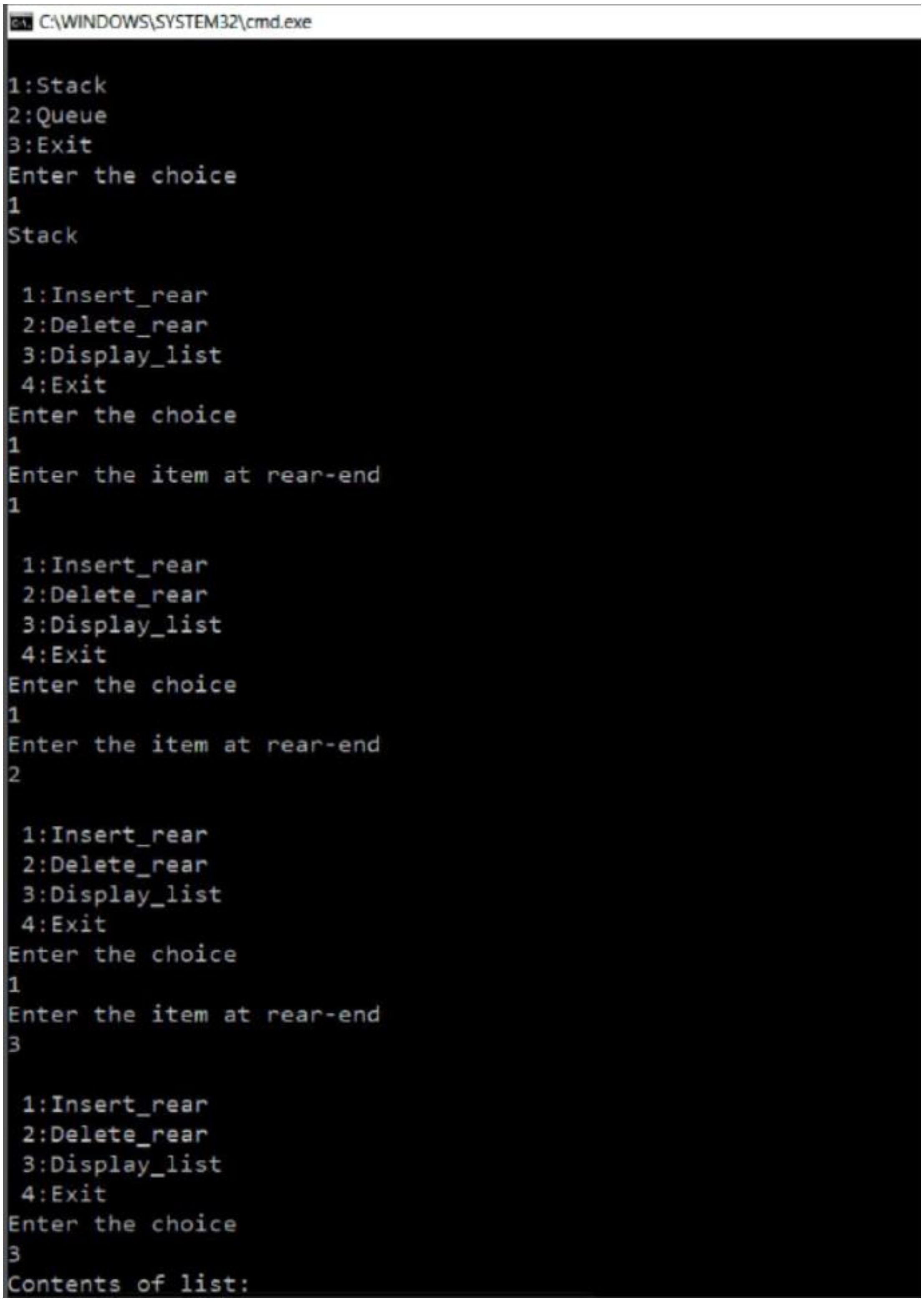
WAP to implement Stack & Queues using Linked Representation .

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

struct node
{
  int info;
  struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("Memory 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)
```

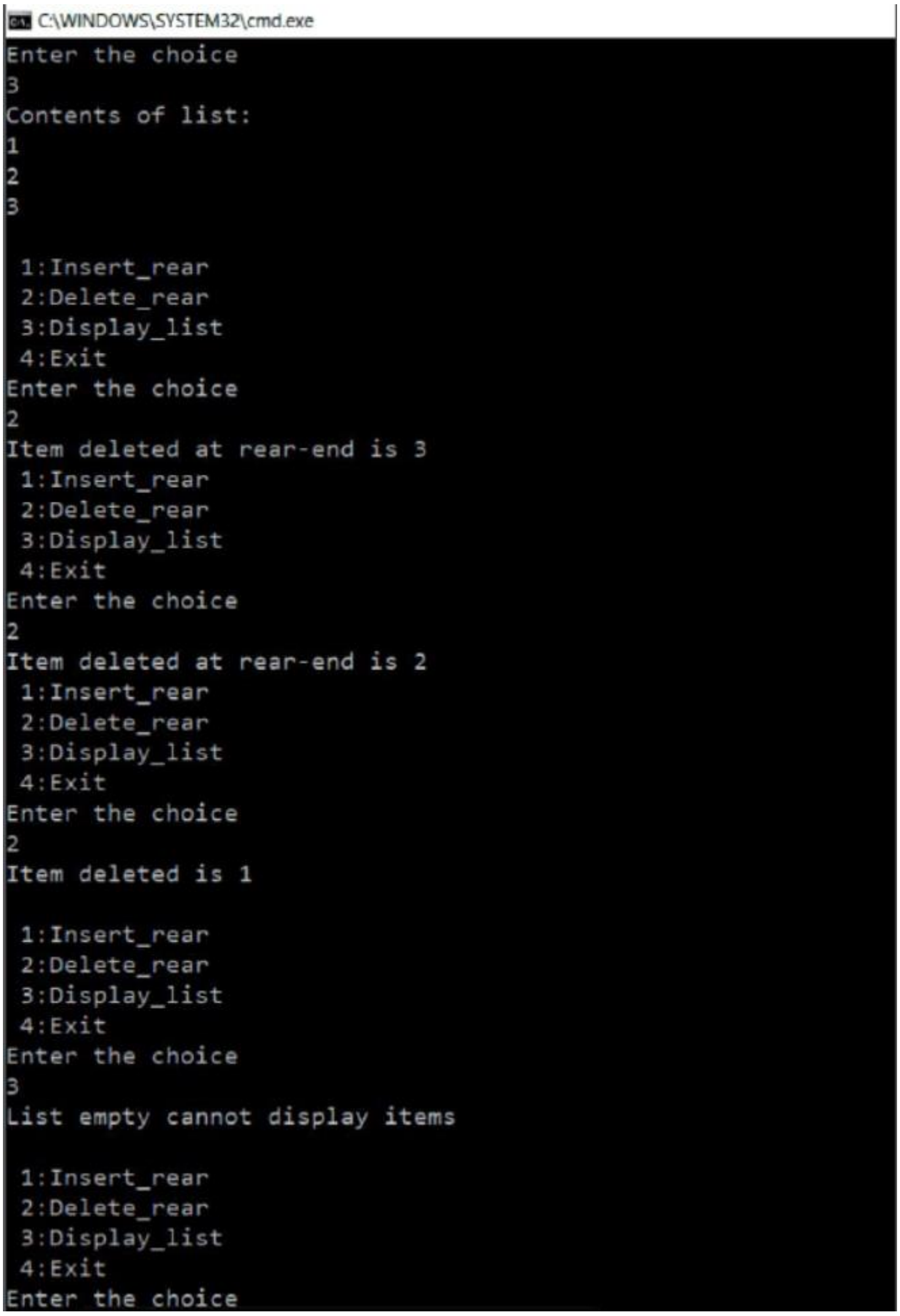
```
return temp;
temp->link=first;
first=temp;
return first;
}

NODE 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;
}
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;
}
```

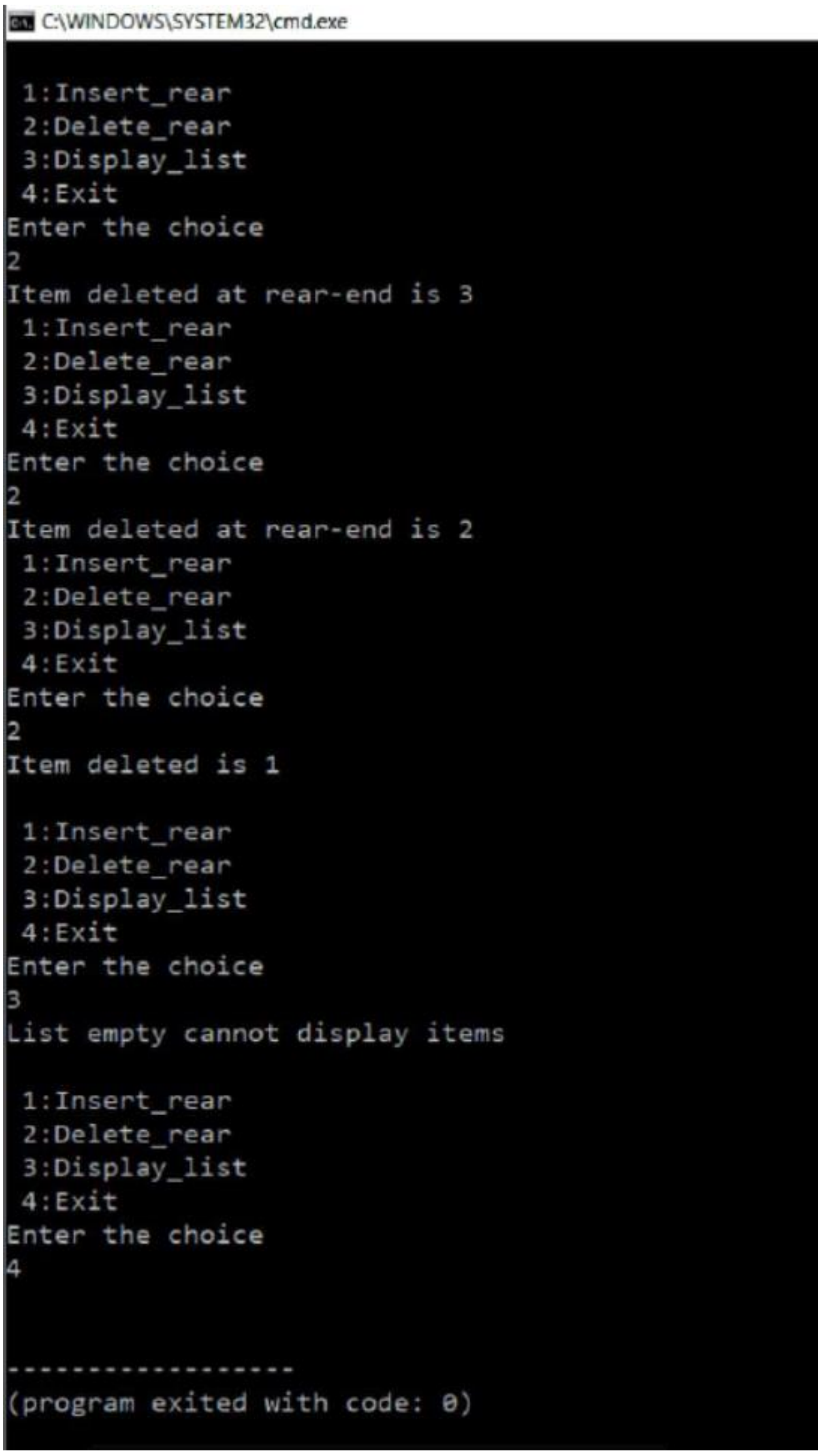
```
void display(NODE first)

{

 NODE temp;

 if(first==NULL)

 {

 printf("List empty cannot display items\n");

 return;

 }

 printf("Contents of list:\n");

 for(temp=first;temp!=NULL;temp=temp->link)

  {

  printf("%d\n",temp->info);

  }

}


int main()

{

int item,choice;

NODE first=NULL,a,b;


for(;;)

{

printf("\n1:Stack\n2:Queue\n3:Exit\n");

printf("Enter the choice\n");

scanf("%d",&choice);

switch(choice)

 {


  case 1:printf("Stack\n");
  for(;;)
  {
   printf("\n 1:Insert_rear\n 2:Delete_rear\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_rear(first);
      break;
   case 3:display(first);
      break;
   default:exit(0);
      break;
   }
  }
case 2:printf("QUEUE\n");
   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;
            }
        }

 case 3:exit(0);
  default:printf("Invalid choice\n");
 }
}
return 0;
}
```



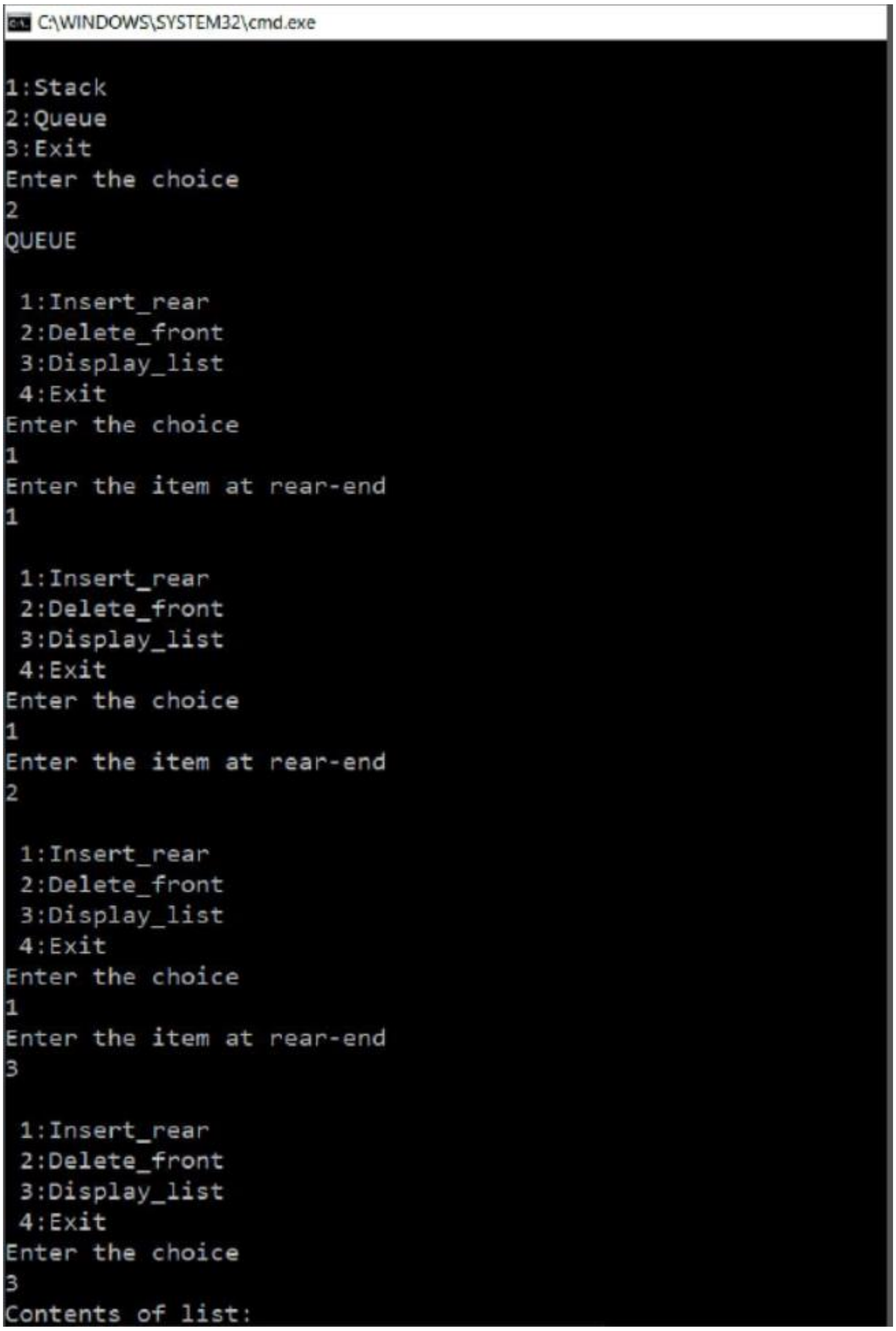
```
C:\WINDOWS\SYSTEM32\cmd.exe

1:Stack
2:Queue
3:Exit
Enter the choice
1
Stack

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

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

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

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
3
Contents of list:
```


C:\WINDOWS\SYSTEM32\cmd.exe

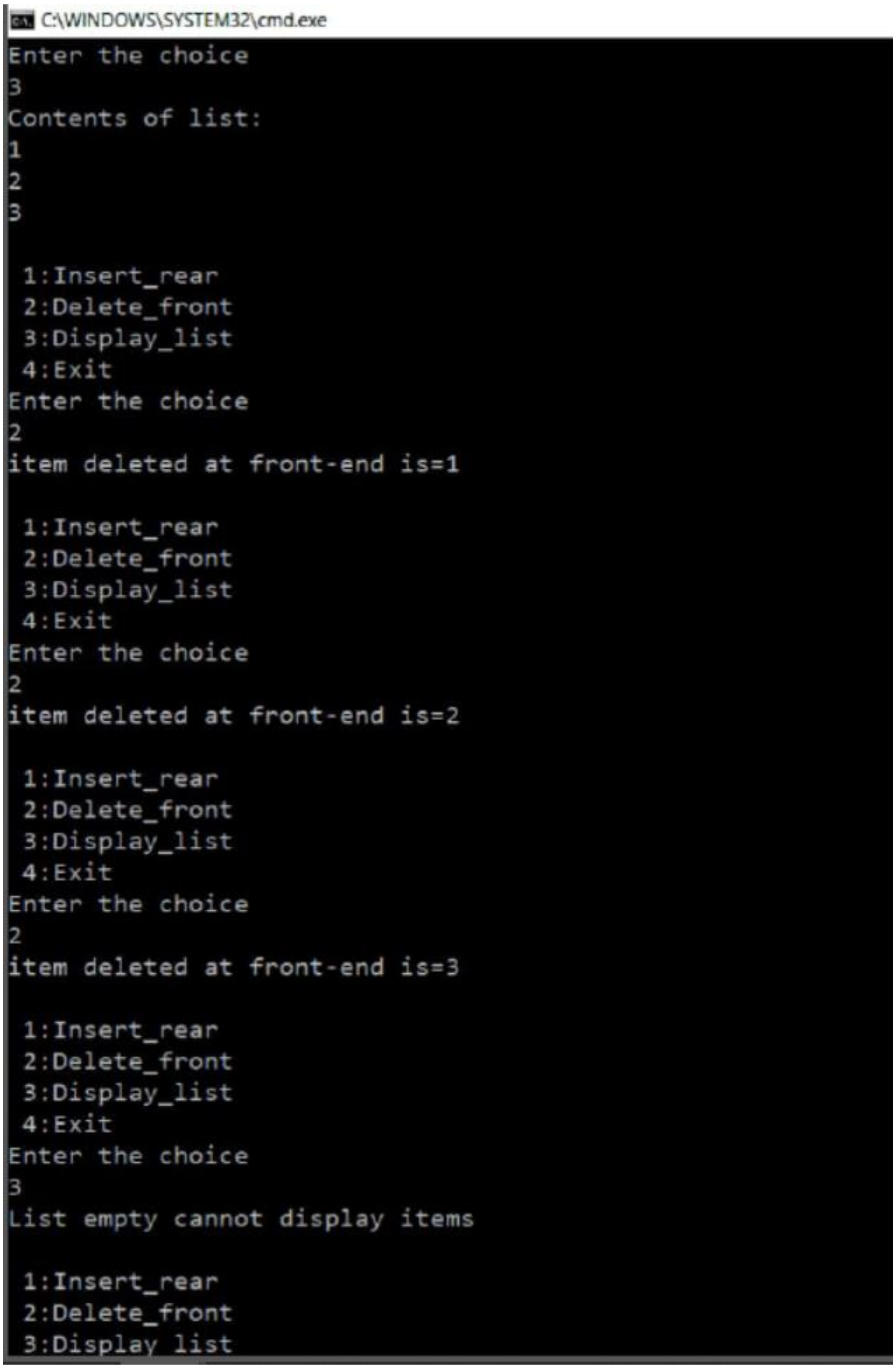
```
Enter the choice
3
Contents of list:
1
2
3

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
2
Item deleted at rear-end is 3
 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
2
Item deleted at rear-end is 2
 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
2
Item deleted is 1

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
3
List empty cannot display items

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



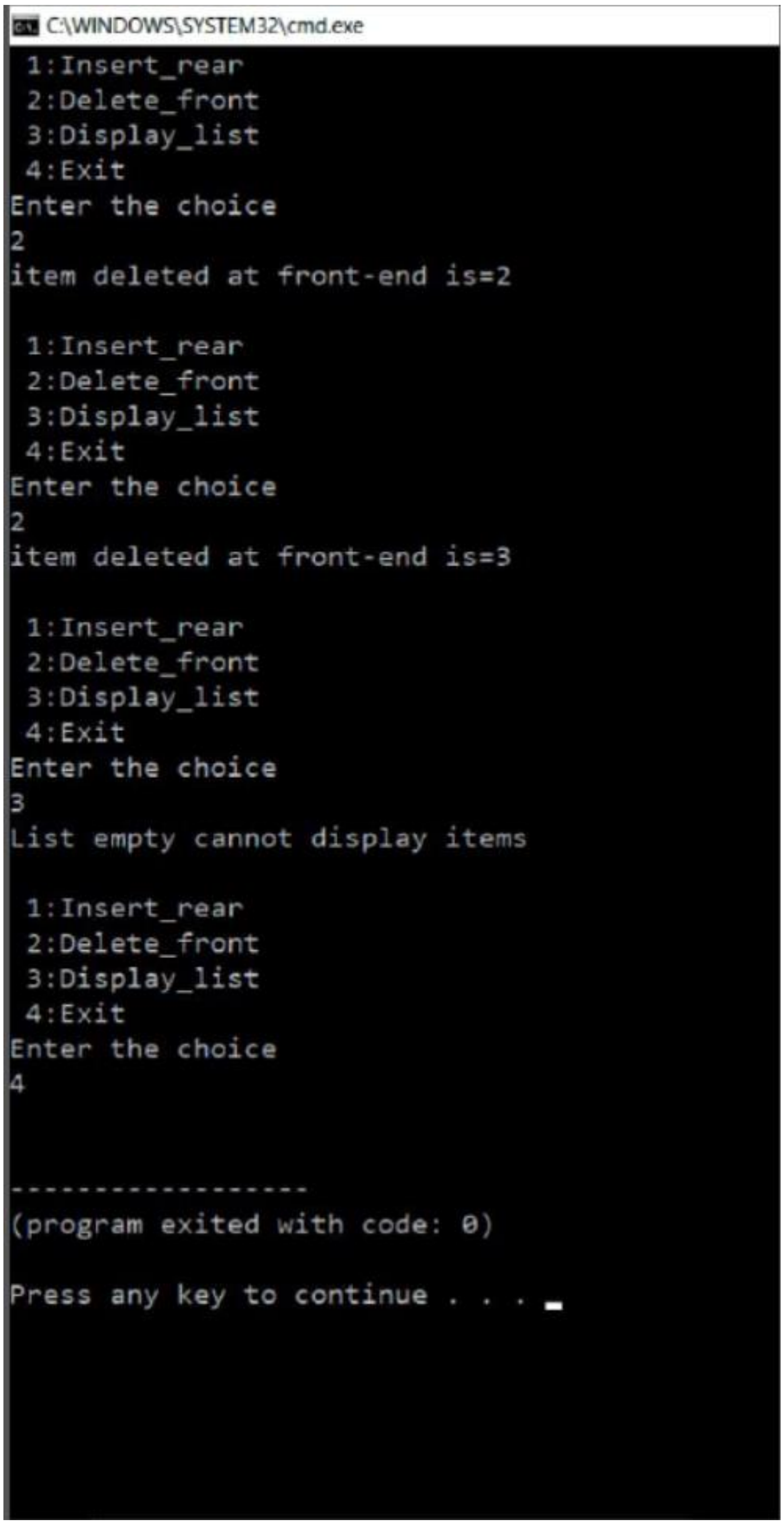
```

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
2
Item deleted at rear-end is 3
 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
2
Item deleted at rear-end is 2
 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
2
Item deleted is 1

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
3
List empty cannot display items

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



-----------------
(program exited with code: 0)
```



```
1:Stack
2:Queue
3:Exit
Enter the choice
2
QUEUE

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

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

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

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
3
Contents of list:
```



```
C:\WINDOWS\SYSTEM32\cmd.exe
Enter the choice
3
Contents of list:
1
2
3

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

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

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

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
3
List empty cannot display items

 1:Insert_rear
 2:Delete_front
 3:Display list
```



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

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

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
3
List empty cannot display items

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



-----------------
(program exited with code: 0)

Press any key to continue . . .
```

LAB PROGRAM 9 :

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.

```
#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;
}
void freenode(NODE x)
{
        free(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)
```

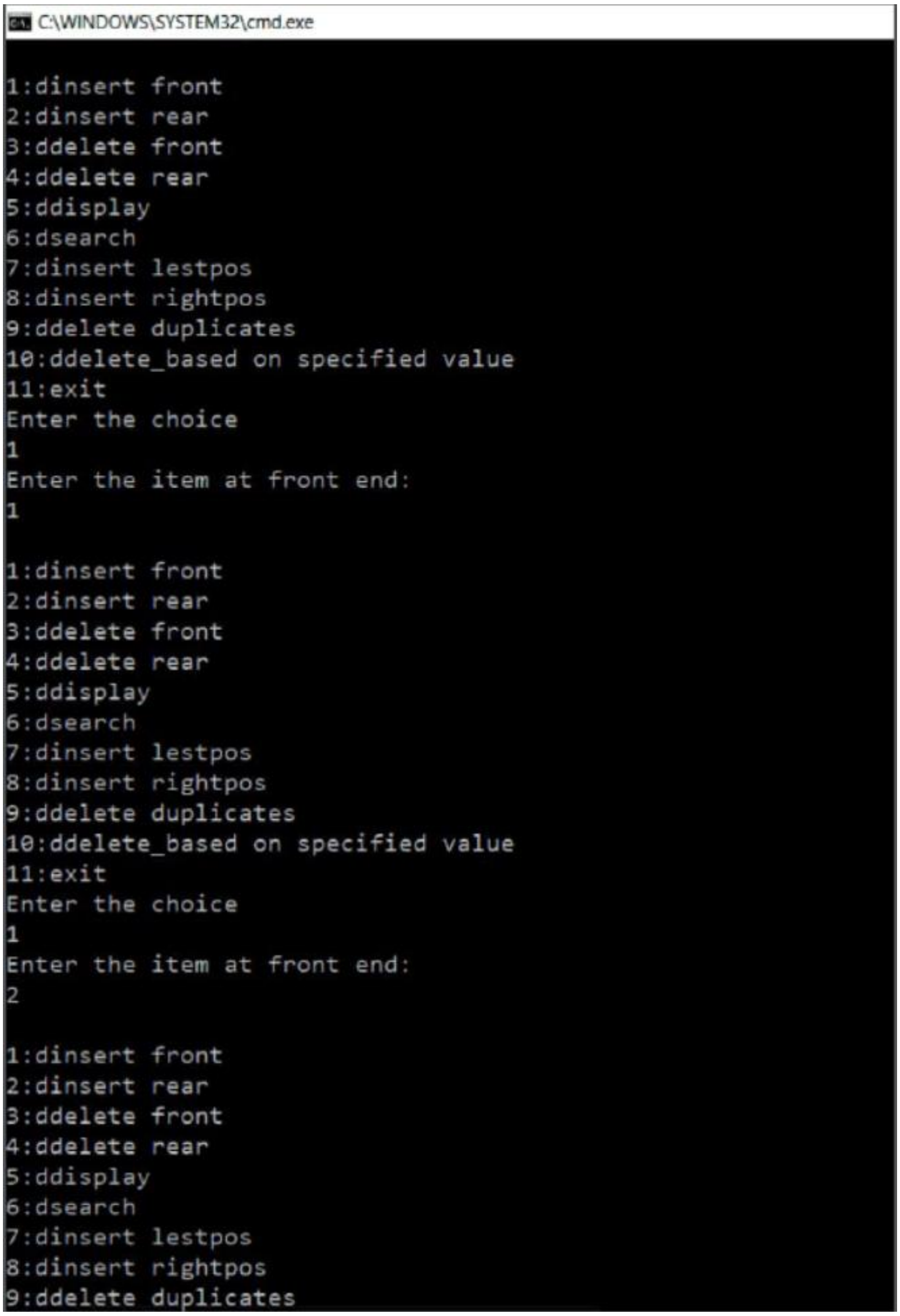
```
{
        NODE cur,prev,temp;
        if (head->rlink==head)
        {
                printf("List is empty\n");
                return head;
        }
        cur=head->rlink;
        while (cur!=head)
        {
                if (cur->info==item)
                {
                        break;
                }
                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("All the duplicate elements of the given item are removed successfully\n");

          }

          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;
}
int main()
{
NODE head;
int item, choice,key;
head=getnode();
head->llink=head;
head->rlink=head;
for(;;)
{
```

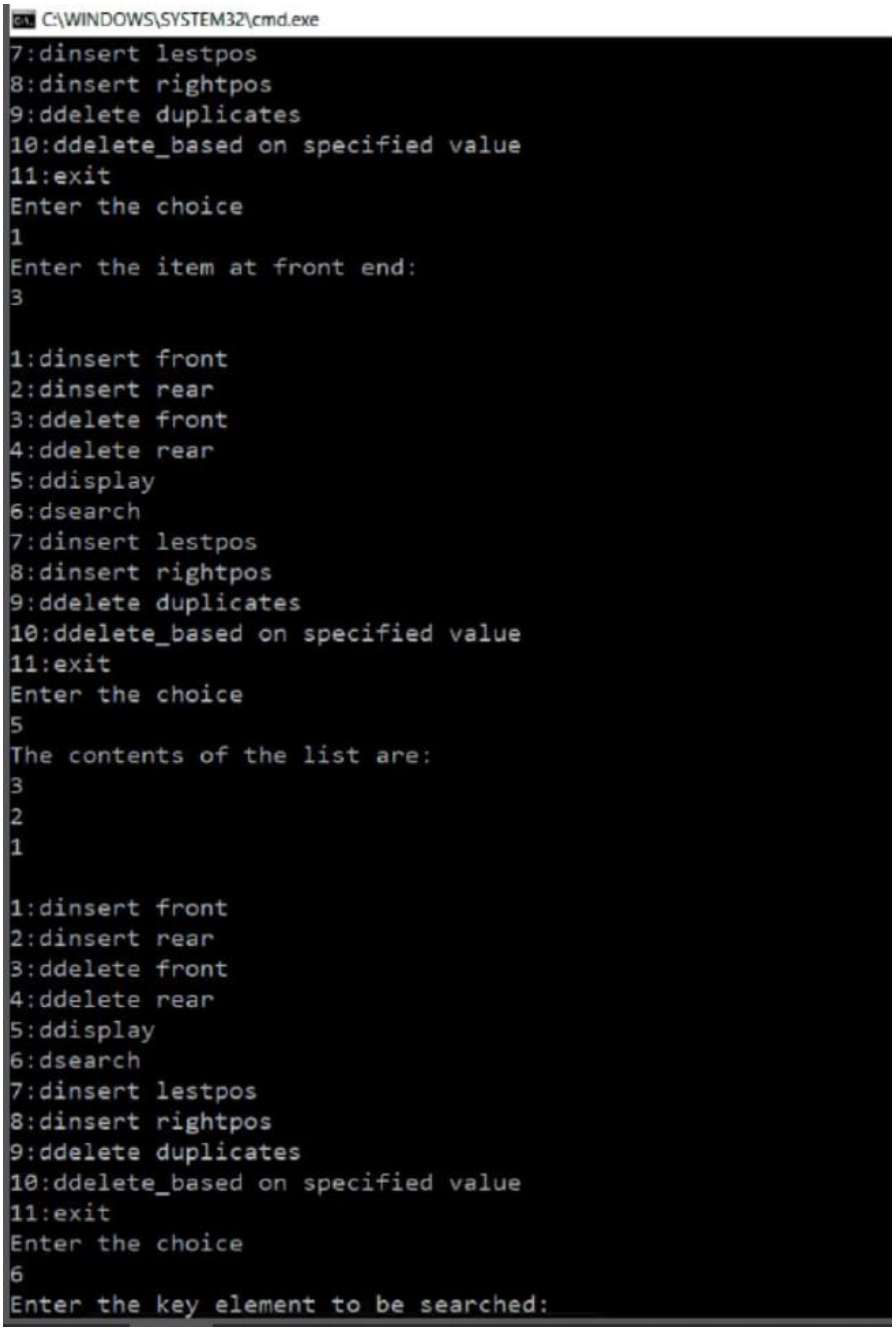
```
            printf("\n1:dinsert front\n2:dinsert rear\n3:ddelete front\n4:ddelete
rear\n5:ddisplay\n6:dsearch\n7:dinsert lestpos\n8:dinsert rightpos\n9:ddelete
duplicates\n10:ddelete_based on specified value\n11:exit\n");

            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;
        case 10:printf("Enter the key value\n");
                     scanf("%d",&item);
                     delete_all_key(item,head);
                     break;
                    case 11:exit(0);
                    default:printf("Invalid choice\n");
                    }
            }
            return 0;
    }
```



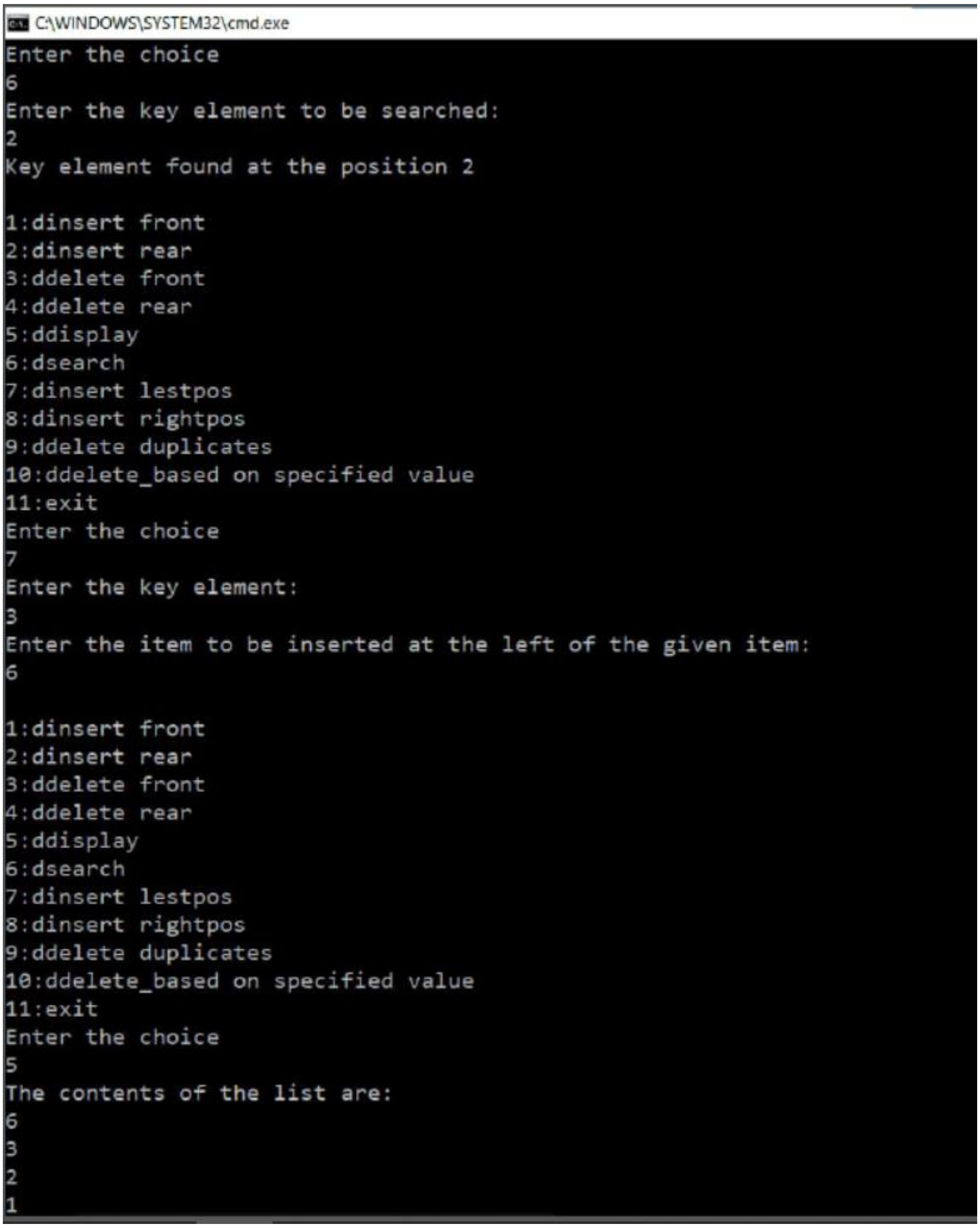
C:\WINDOWS\SYSTEM32\cmd.exe

```
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:ddelete_based on specified value
11: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:ddelete_based on specified value
11:exit
Enter the choice
1
Enter the item at front end:
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
```

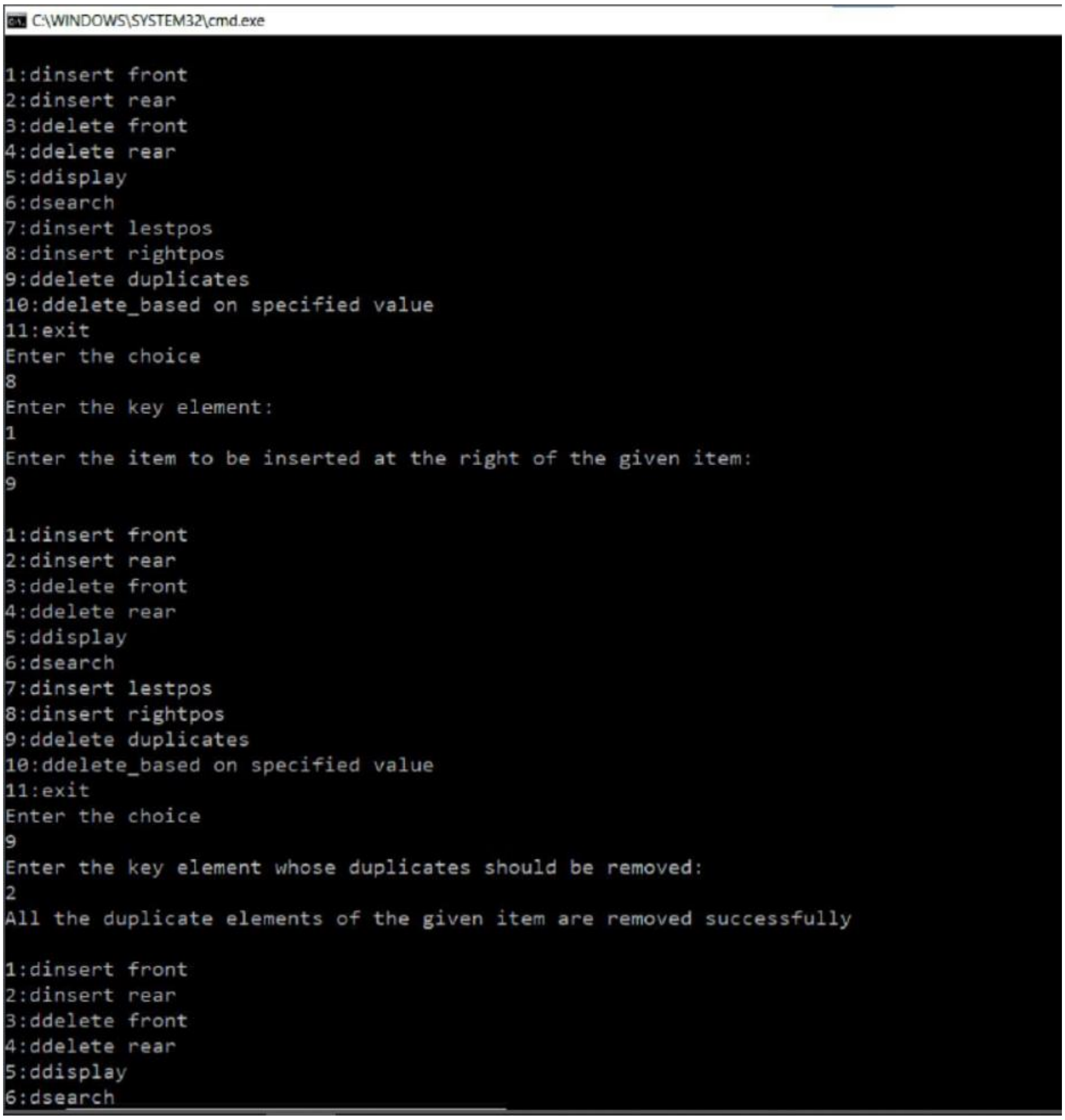


C:\WINDOWS\SYSTEM32\cmd.exe

```
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
1
Enter the item at front end:
3

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:ddelete_based on specified value
11:exit
Enter the choice
5
The contents of the list are:
3
2
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:ddelete_based on specified value
11:exit
Enter the choice
6
Enter the key element to be searched:
```

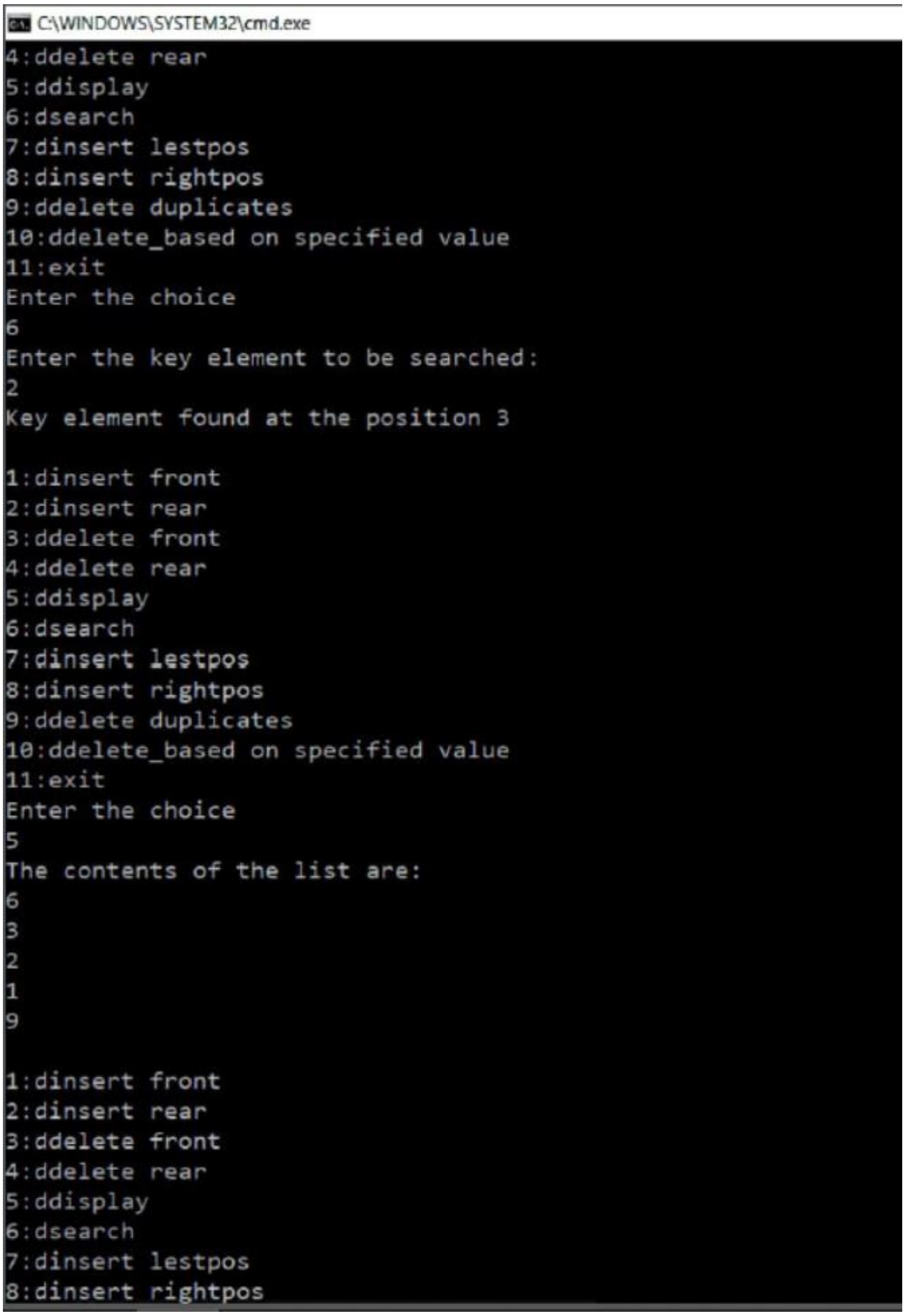

C:\WINDOWS\SYSTEM32\cmd.exe

```
Enter the choice
6
Enter the key element to be searched:
2
Key element found at the position 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:ddelete_based on specified value
11:exit
Enter the choice
7
Enter the key element:
3
Enter the item to be inserted at the left of the given item:
6

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:ddelete_based on specified value
11:exit
Enter the choice
5
The contents of the list are:
6
3
2
1
```

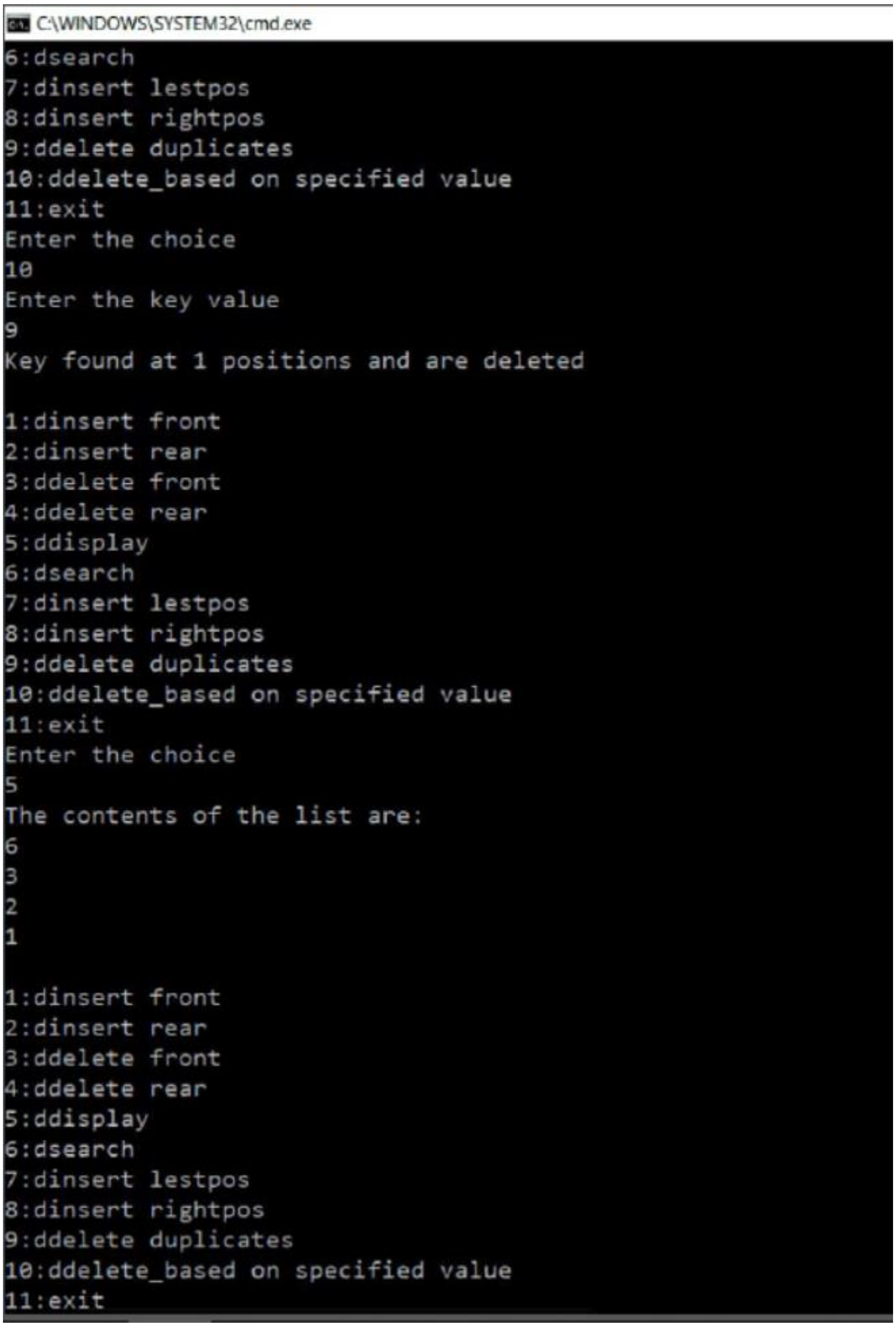


C:\WINDOWS\SYSTEM32\cmd.exe

```
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:ddelete_based on specified value
11:exit
Enter the choice
8
Enter the key element:
1
Enter the item to be inserted at the right of the given item:
9

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:ddelete_based on specified value
11:exit
Enter the choice
9
Enter the key element whose duplicates should be removed:
2
All the duplicate elements of the given item are removed successfully

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
```



```
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
6
Enter the key element to be searched:
2
Key element found at the position 3

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:ddelete_based on specified value
11:exit
Enter the choice
5
The contents of the list are:
6
3
2
1
9

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
```



```
C:\WINDOWS\SYSTEM32\cmd.exe
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
10
Enter the key value
9
Key found at 1 positions and are deleted

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:ddelete_based on specified value
11:exit
Enter the choice
5
The contents of the list are:
6
3
2
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:ddelete_based on specified value
11:exit
```

LAB PROGRAM 10 :

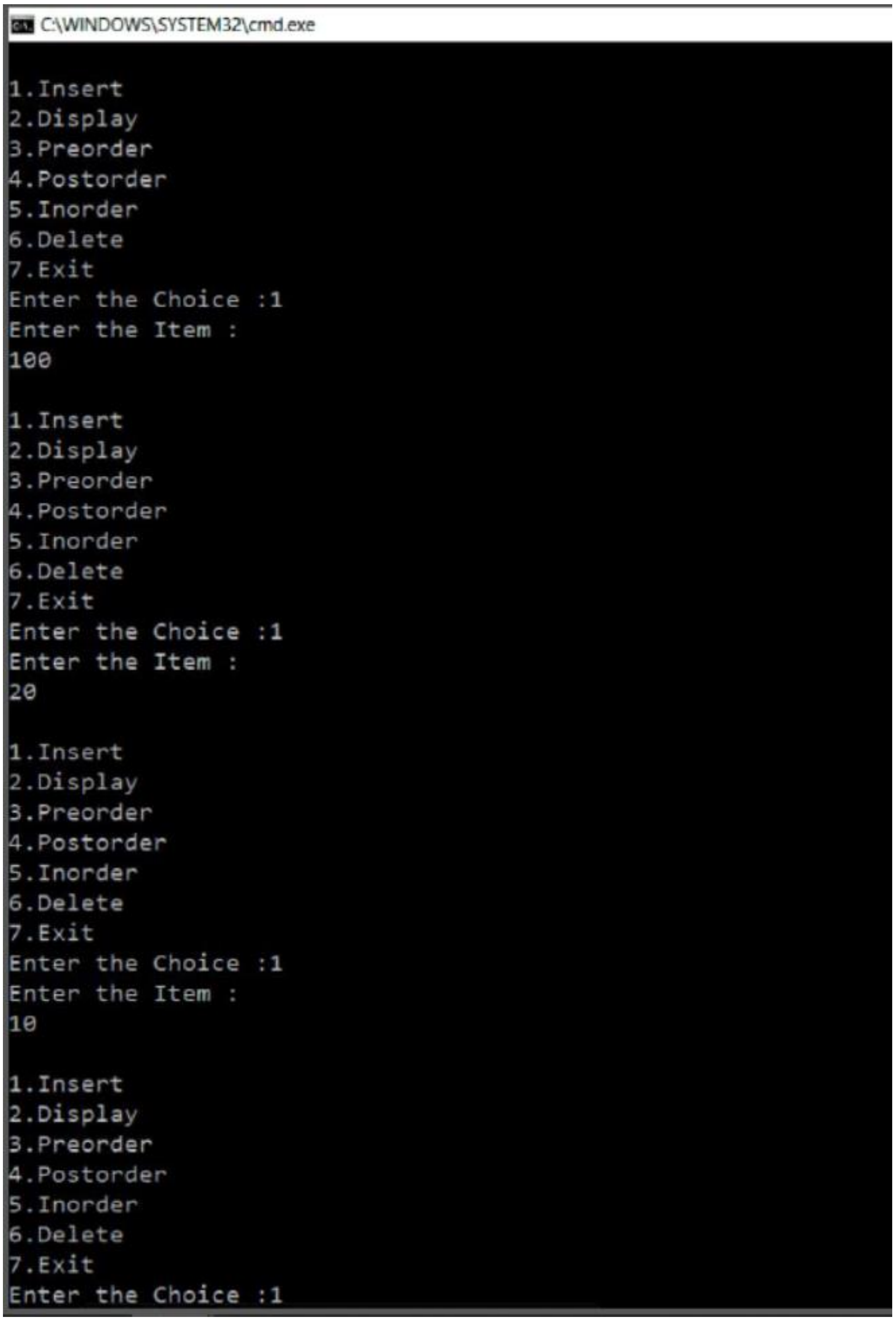
Write a program a) To construct a binary Search tree. b) To traverse the tree using all the methods i.e., in-order, preorder and post order c) To display the elements in the tree.

```
#include<stdio.h>
#include<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("Memory 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;
```

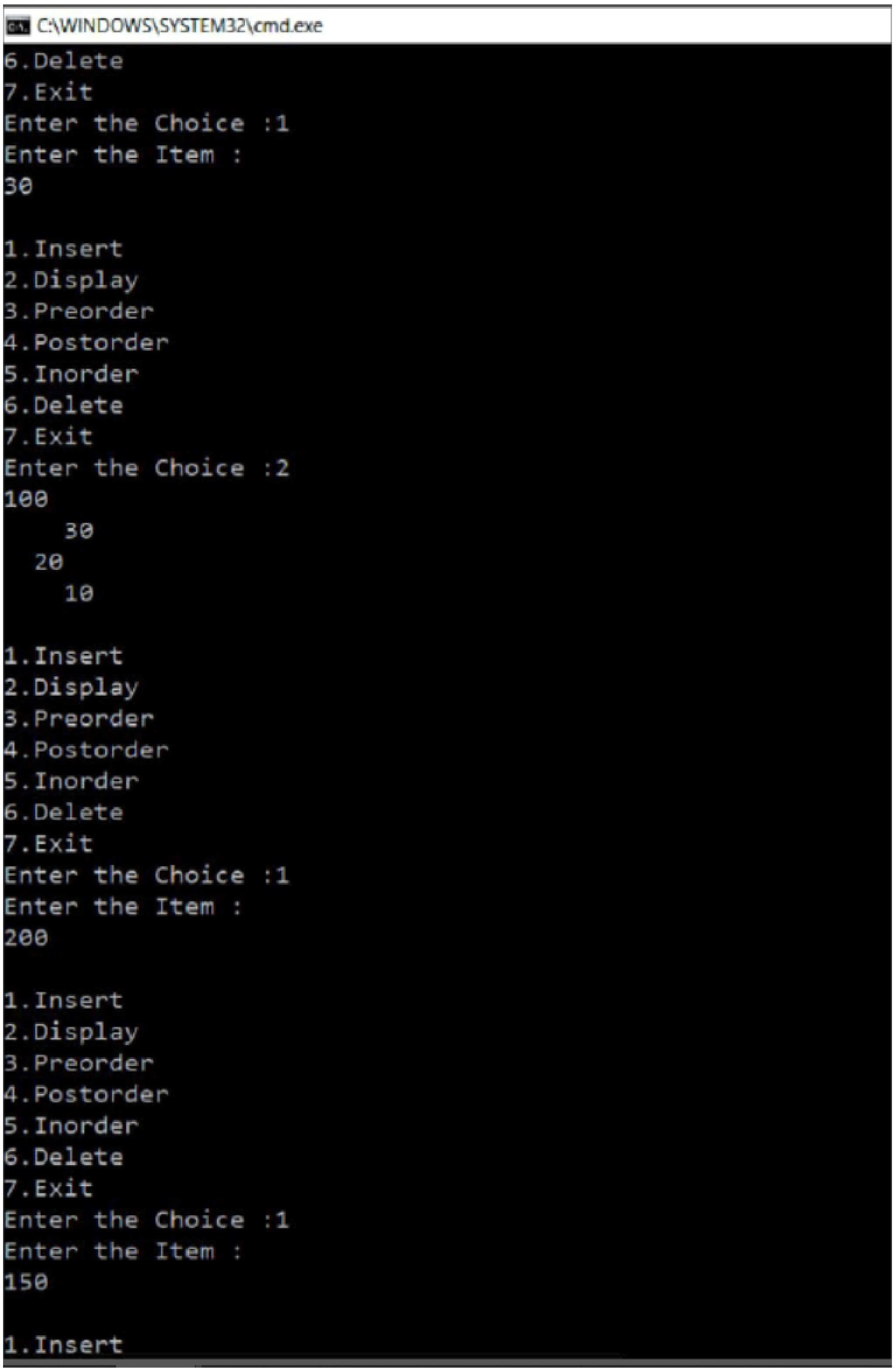
```
temp->info=item;
if(root==NULL)
 return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
 prev->llink=temp;
else
 prev->rlink=temp;
return root;
}
void display(NODE root,int i)
{
int j;
if(root!=NULL)
 {
  display(root->rlink,i+1);
  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);
  }
 }
int main()
{
int item,choice;
NODE root=NULL;
for(;;)
{
printf("\n1.Insert\n2.Display\n3.Preorder\n4.Postorder\n5.Inorder\n6.Delete\n7.Exit\n");
printf("Enter the Choice :");
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);
```

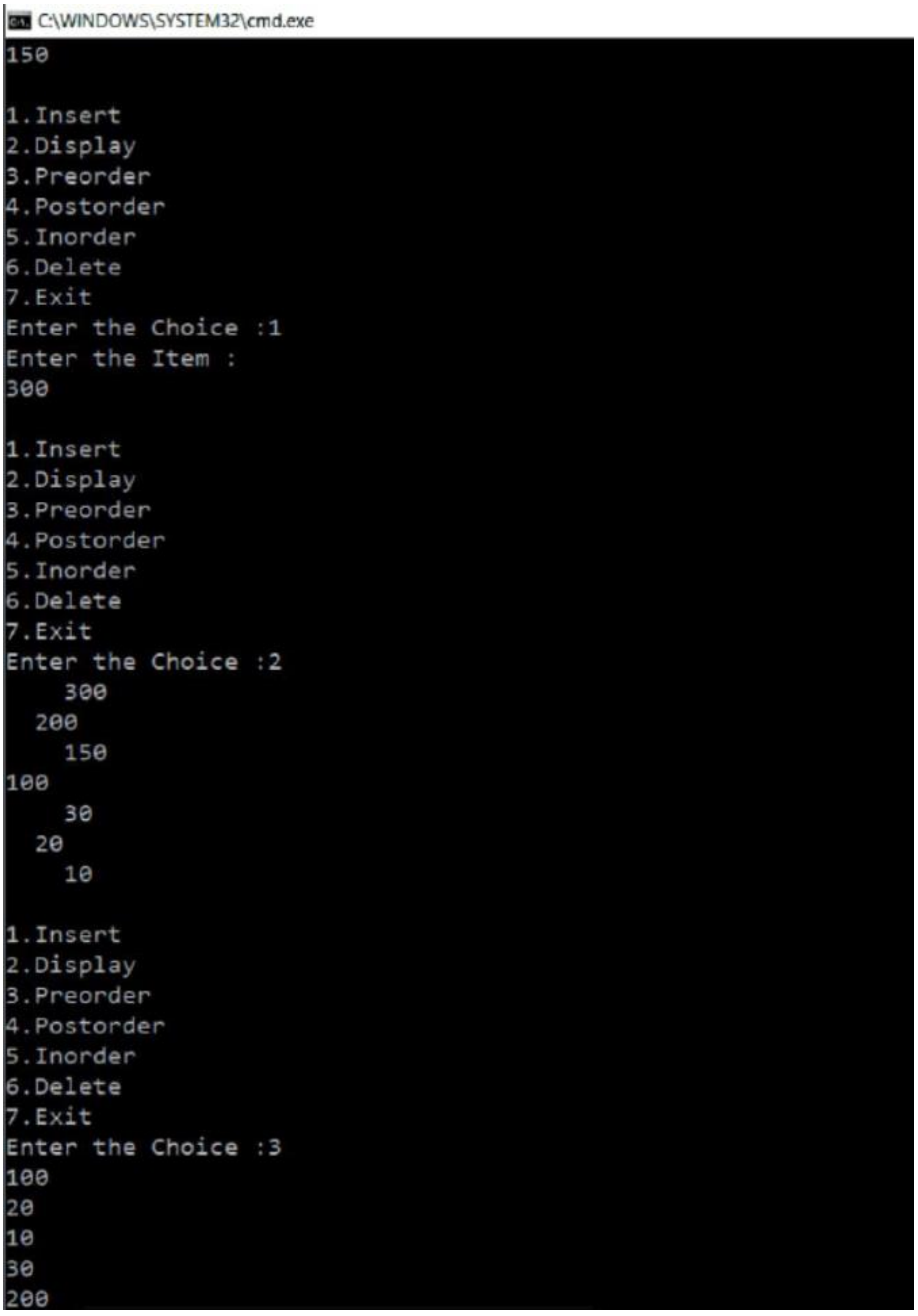
```
                          root=delete(root,item);

                          break;

 default:exit(0);

                           break;

              }

             }

             return 0;

}
```


C:\WINDOWS\SYSTEM32\cmd.exe

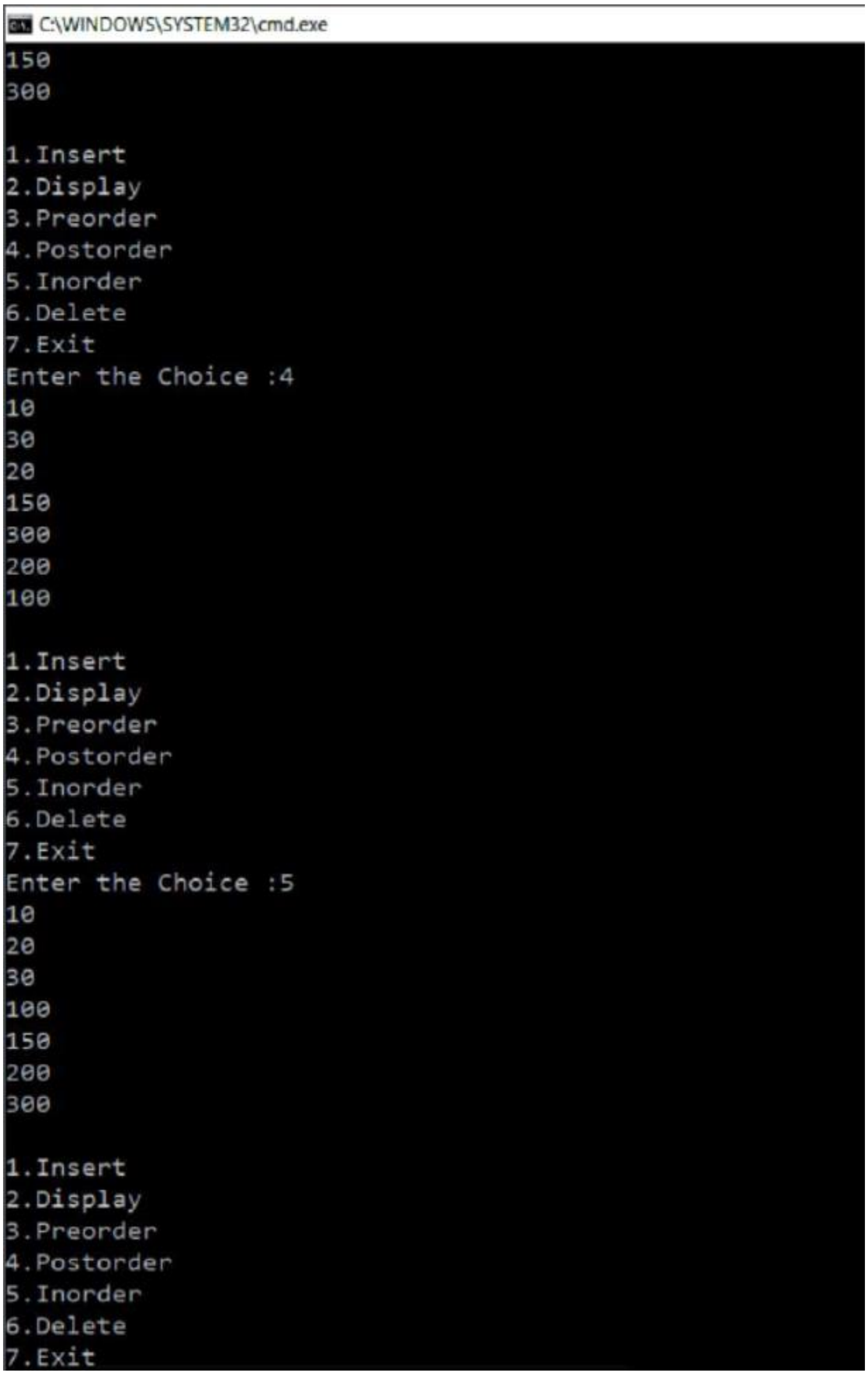
```
1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :1
Enter the Item :
100

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :1
Enter the Item :
20

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :1
Enter the Item :
10

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :1
```



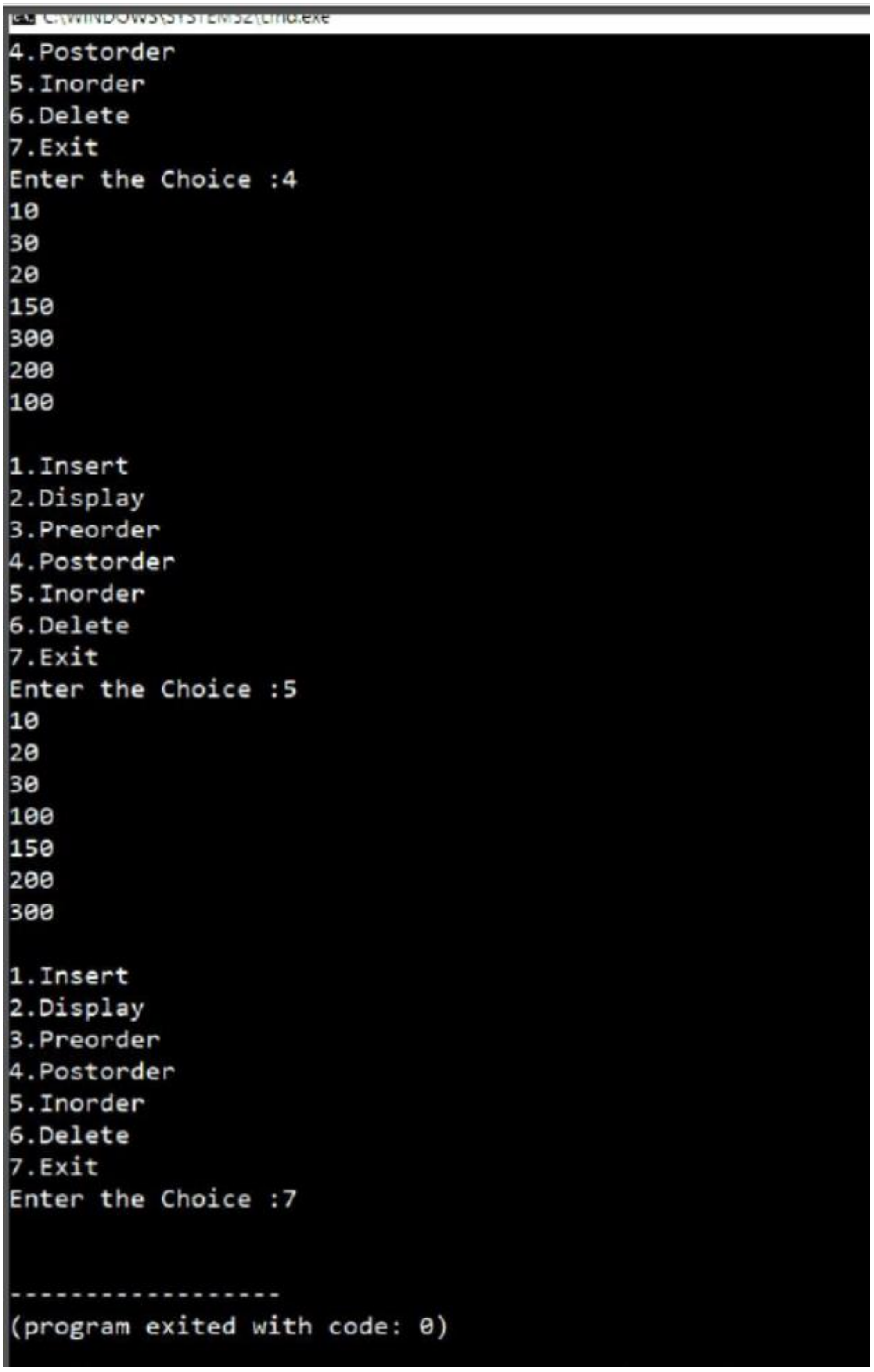
```
6.Delete
7.Exit
Enter the Choice :1
Enter the Item :
30

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :2
100
    30
  20
    10

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :1
Enter the Item :
200

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :1
Enter the Item :
150

1.Insert
```


C:\WINDOWS\SYSTEM32\cmd.exe

```
150

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :1
Enter the Item :
300

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :2
    300
  200
    150
100
    30
  20
    10

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :3
100
20
10
30
200
```



```
150
300

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :4
10
30
20
150
300
200
100

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :5
10
20
30
100
150
200
300

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
```



```
C:\WINDOWS\SYSTEM32\cmd.exe
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :4
10
30
20
150
300
200
100

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :5
10
20
30
100
150
200
300

1.Insert
2.Display
3.Preorder
4.Postorder
5.Inorder
6.Delete
7.Exit
Enter the Choice :7


------------------
(program exited with code: 0)
```