LAB REPORT

NAME:NITHIN.C

USN:1BM19CS106

SUBJECT:DATA STRUCTURES

ACADEMIC YEAR:2020-21

BATCH:3

Lab Program 1:

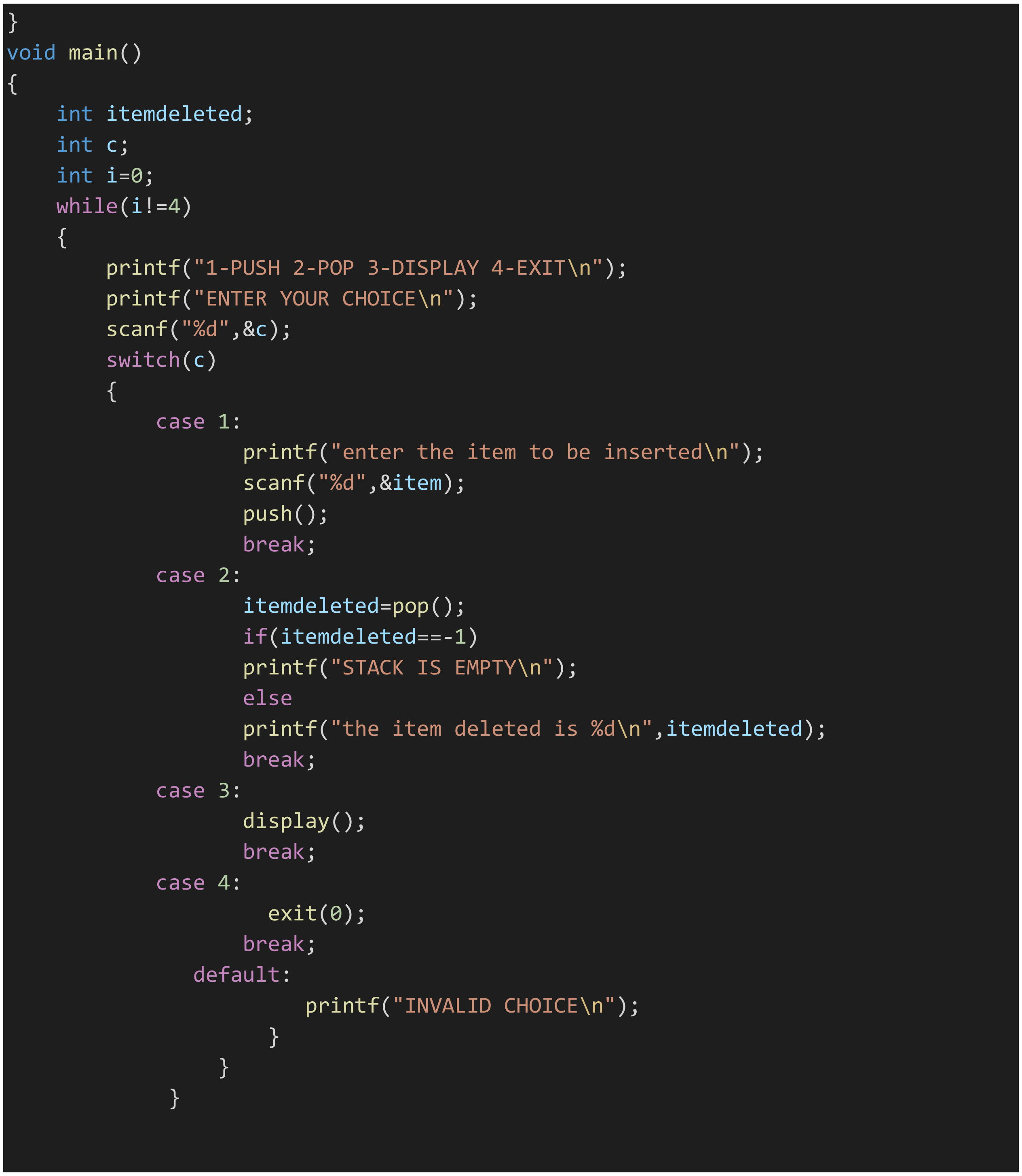
Write a program to simulate the working of stack using an array with the following :

a) Push

b) Pop

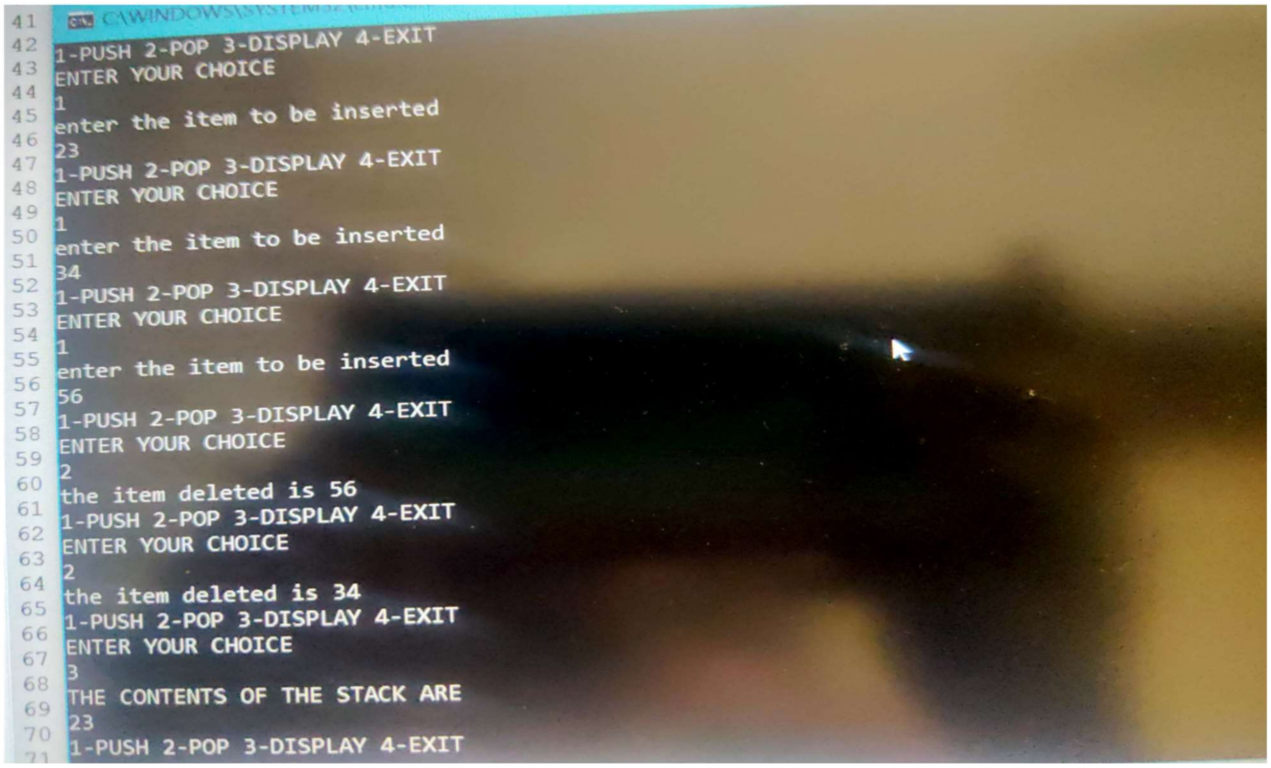
c)Display

```c
#include<stdio.h>
#include<process.h>
#define stacksize 5
int top=-1;
int s[10];
int item;
void push()
{
    if(top==(stacksize-1))
    {
        printf("STACK OVERFLOW\n");
          return;
      }
      top=top+1;
      s[top]=item;
  }
  int pop()
  {
      if(top==-1)
      return -1;
     return  s[top--];
  }
  void display()
  {
      int i;
       if(top==-1)
       {
           printf("THE STACK IS EMPTY/STACK UNDERFLOW\n");
           return;
       }

printf("THE CONTENTS OF THE STACK ARE\n");
for(i=0;i<=top;i++)
{
    printf("%d\n",s[i]);
}
```

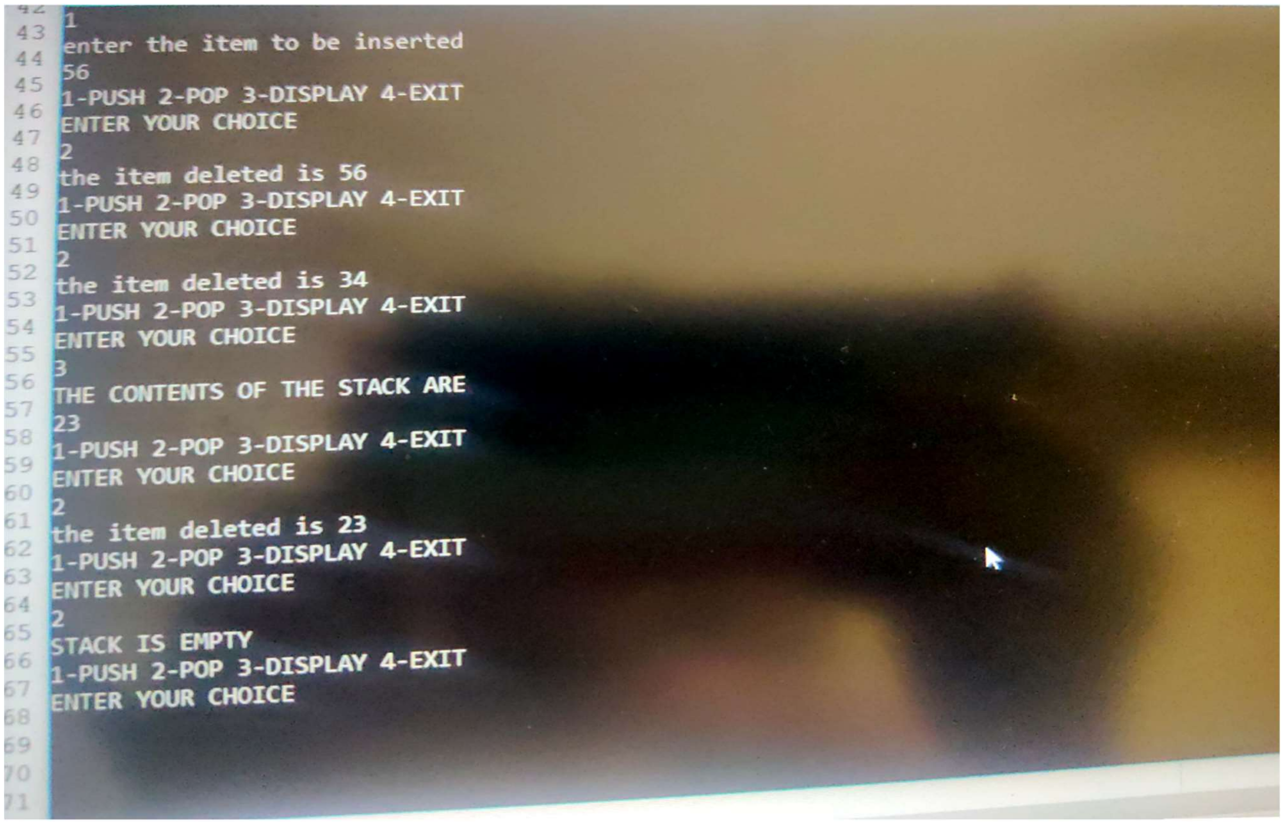


```c
}
void main()
{
    int itemdeleted;
    int c;
    int i=0;
    while(i!=4)
    {
        printf("1-PUSH 2-POP 3-DISPLAY 4-EXIT\n");
        printf("ENTER YOUR CHOICE\n");
        scanf("%d",&c);
        switch(c)
        {
            case 1:
                   printf("enter the item to be inserted\n");
                   scanf("%d",&item);
                   push();
                   break;
            case 2:
                   itemdeleted=pop();
                   if(itemdeleted==-1)
                   printf("STACK IS EMPTY\n");
                   else
                   printf("the item deleted is %d\n",itemdeleted);
                   break;
            case 3:
                   display();
                   break;
            case 4:
                     exit(0);
                   break;
               default:
                        printf("INVALID CHOICE\n");
                     }
                 }
             }
```

Output:



```
C:\WINDOWS\SYSTEM32\
1-PUSH 2-POP 3-DISPLAY 4-EXIT
ENTER YOUR CHOICE
1
enter the item to be inserted
23
1-PUSH 2-POP 3-DISPLAY 4-EXIT
ENTER YOUR CHOICE
1
enter the item to be inserted
34
1-PUSH 2-POP 3-DISPLAY 4-EXIT
ENTER YOUR CHOICE
1
enter the item to be inserted
56
1-PUSH 2-POP 3-DISPLAY 4-EXIT
ENTER YOUR CHOICE
2
the item deleted is 56
1-PUSH 2-POP 3-DISPLAY 4-EXIT
ENTER YOUR CHOICE
2
the item deleted is 34
1-PUSH 2-POP 3-DISPLAY 4-EXIT
ENTER YOUR CHOICE
3
THE CONTENTS OF THE STACK ARE
23
1-PUSH 2-POP 3-DISPLAY 4-EXIT
```



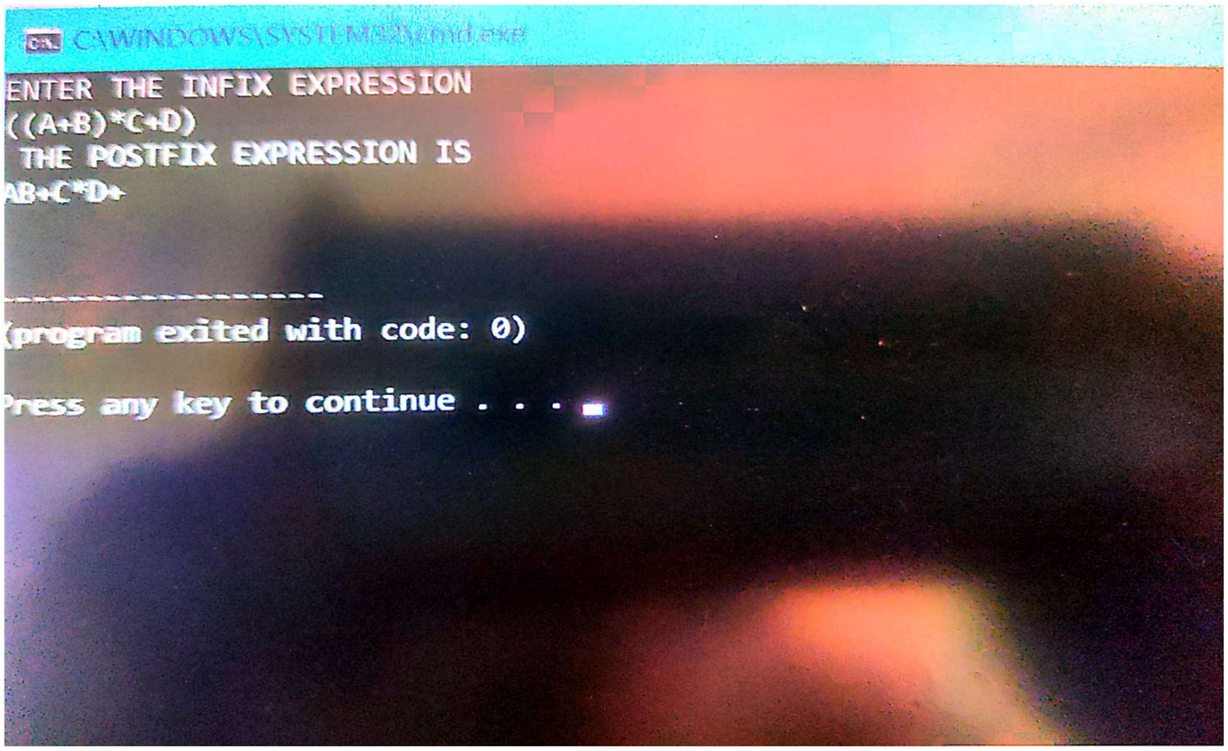
```
1
enter the item to be inserted
56
1-PUSH 2-POP 3-DISPLAY 4-EXIT
ENTER YOUR CHOICE
2
the item deleted is 56
1-PUSH 2-POP 3-DISPLAY 4-EXIT
ENTER YOUR CHOICE
2
the item deleted is 34
1-PUSH 2-POP 3-DISPLAY 4-EXIT
ENTER YOUR CHOICE
3
THE CONTENTS OF THE STACK ARE
23
1-PUSH 2-POP 3-DISPLAY 4-EXIT
ENTER YOUR CHOICE
2
the item deleted is 23
1-PUSH 2-POP 3-DISPLAY 4-EXIT
ENTER YOUR CHOICE
2
STACK IS EMPTY
1-PUSH 2-POP 3-DISPLAY 4-EXIT
ENTER YOUR CHOICE
```

Lab Program 2:

Write a program 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)

```c
#include<stdio.h>
#include<process.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 infixtopostfix(char infix[],char postfix[])
{
    int i,j;
    j=0;
    int top=-1;
    char s[50];
    char symbol;
    s[++top]='#';
    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='\0';
}
int main()
{
    char infix[50];
    char postfix[50];
    printf("ENTER THE INFIX EXPRESSION\n");
    scanf("%s",infix);
    infixtopostfix(infix,postfix);
    printf(" THE POSTFIX EXPRESSION IS\n");
    printf("%s\n",postfix);
}
```

Output:



```
C:\WINDOWS\SYSTEM32\cmd.exe
ENTER THE INFIX EXPRESSION
((A+B)*C+D)
 THE POSTFIX EXPRESSION IS
AB+C*D+

------------------
(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

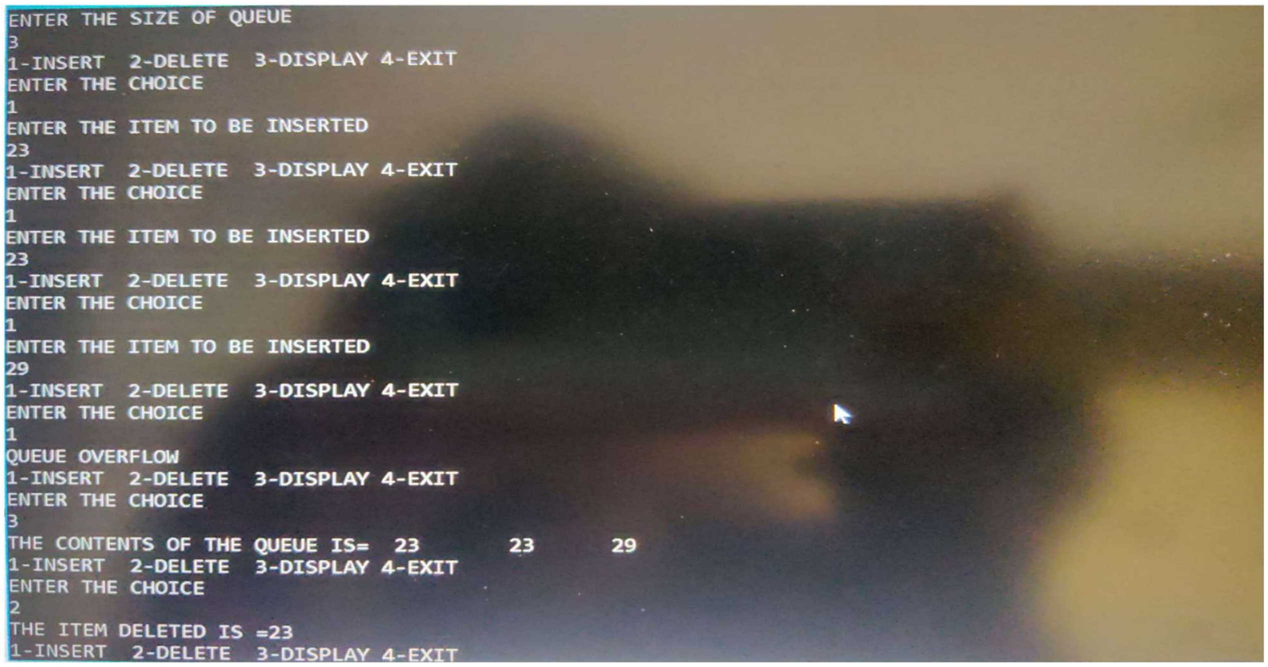
b) Delete

c) Display

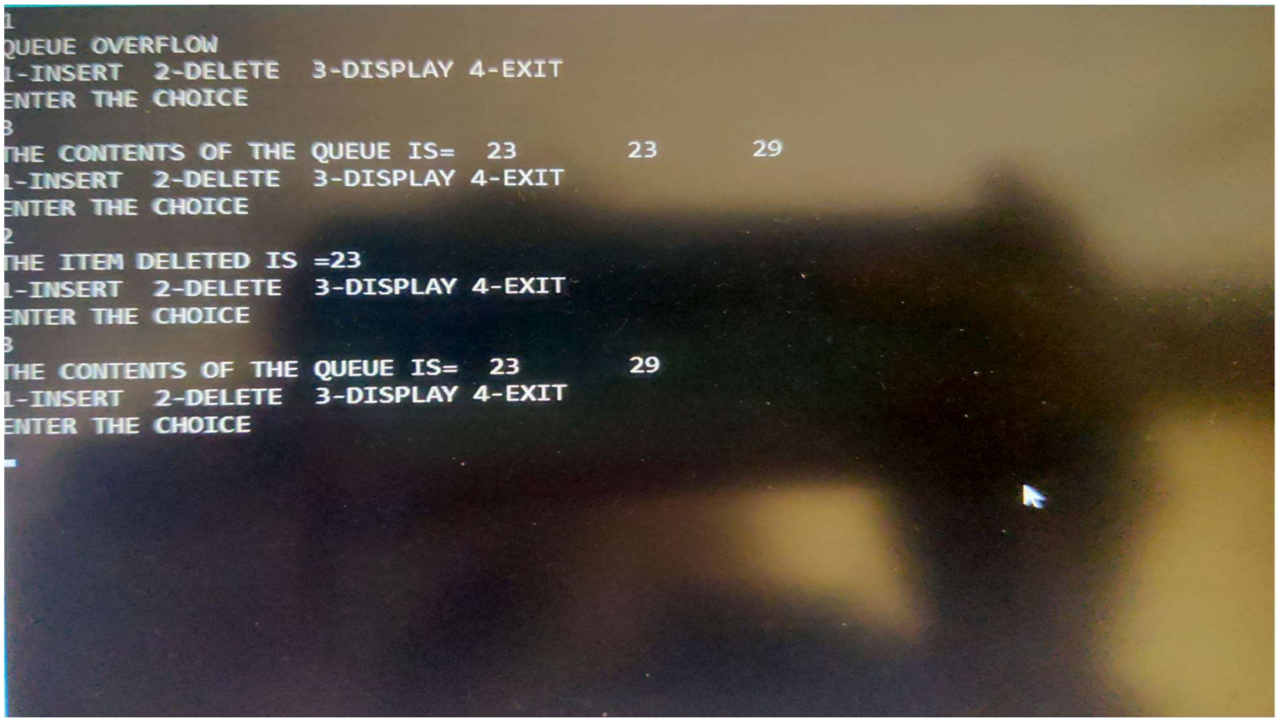
```
#include<stdio.h>
#include<conio.h>
#include<process.h>
int item;
int r=-1;
int f=0;
int n;
int q[10];
void insertrear()
{
```

```c
    if(r==(n-1))
    {
        printf("QUEUE OVERFLOW\n");
        return;
    }
    printf("ENTER THE ITEM TO BE INSERTED\n");
    scanf("%d",&item);
    r=r+1;
    q[r]=item;
}
void deletefront()
{
    if(f>r)
    {
        printf("THE QUEUE IS EMPTY\n");
        f=0;
        r=-1;
    }
    else
    printf("THE ITEM DELETED IS =%d\n",q[f++]);
}
void display()
{
    int i;
    if(f>r)
    {
        printf("THE QUEUE IS EMPTY\n");
        return;
    }
    else
    {
    printf("THE CONTENTS OF THE QUEUE IS=  ");
    for(i=f;i<=r;i++)
    {
        printf("%d\t",q[i]);}
    printf("\n");
}
}
int main()
{
    int c,i;
    printf("ENTER THE SIZE OF QUEUE\n");
    scanf("%d",&n);
    while(i!=4)
    {
        printf("1-INSERT  2-DELETE  3-DISPLAY 4-EXIT\n");
        printf("ENTER THE CHOICE\n");
        scanf("%d",&c);
        switch(c)
        {
            case 1:
                    insertrear();
                    break;
            case 2:
                    deletefront();
                    break;
            case 3:
                    display();
                    break;
            case 4:
                    exit(0);
          default:
                    printf("INVALID CHOICE");
                }
            }
        }
```

Output:



```
ENTER THE SIZE OF QUEUE
3
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
1
ENTER THE ITEM TO BE INSERTED
23
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
1
ENTER THE ITEM TO BE INSERTED
23
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
1
ENTER THE ITEM TO BE INSERTED
29
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
1
QUEUE OVERFLOW
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
3
THE CONTENTS OF THE QUEUE IS=  23       23       29
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
2
THE ITEM DELETED IS =23
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
```



```
1
QUEUE OVERFLOW
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
3
THE CONTENTS OF THE QUEUE IS=  23       23       29
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
2
THE ITEM DELETED IS =23
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
3
THE CONTENTS OF THE QUEUE IS=  23       29
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
```

Lab Program 4:

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

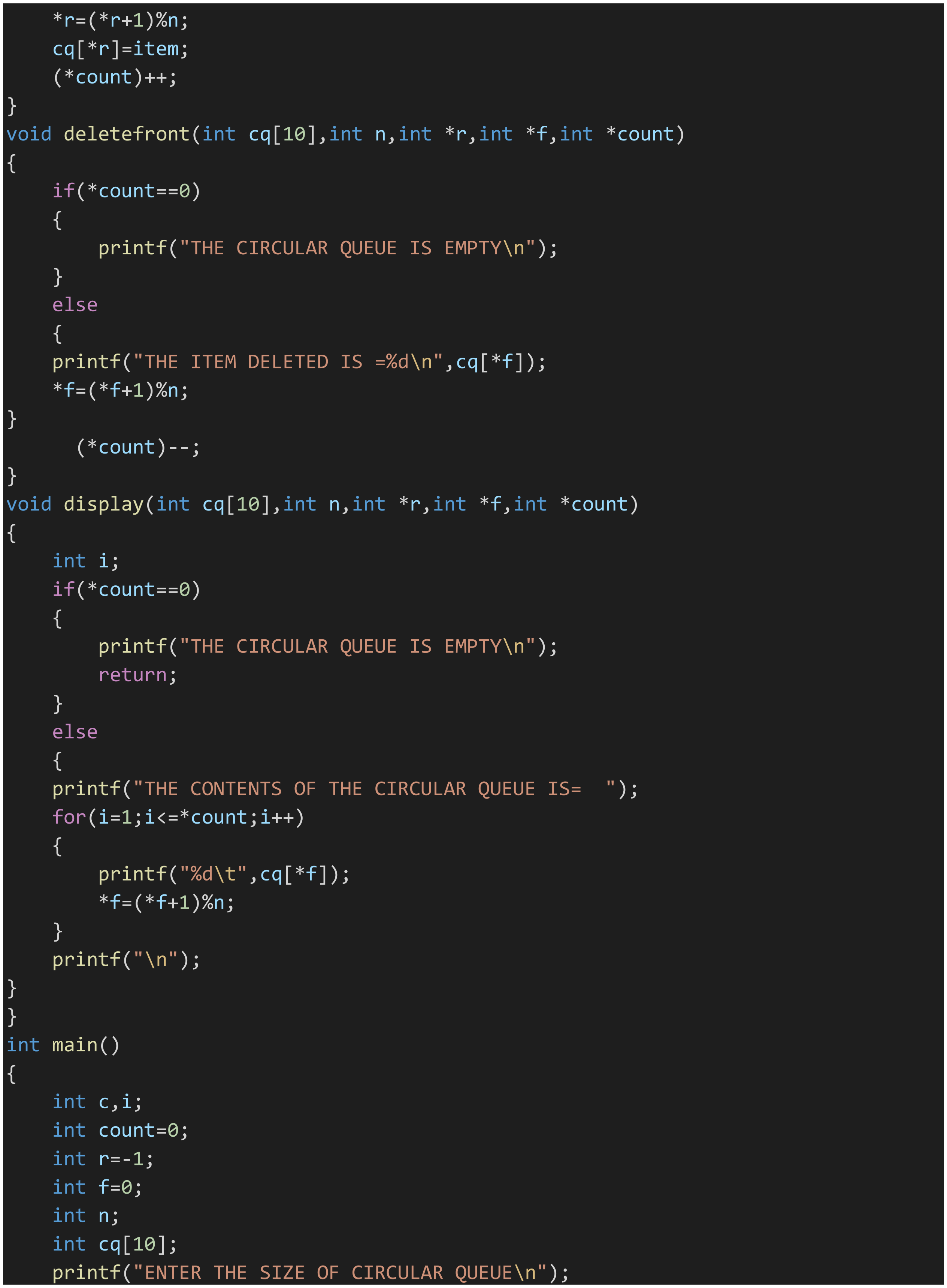
a) Insert

b) Delete

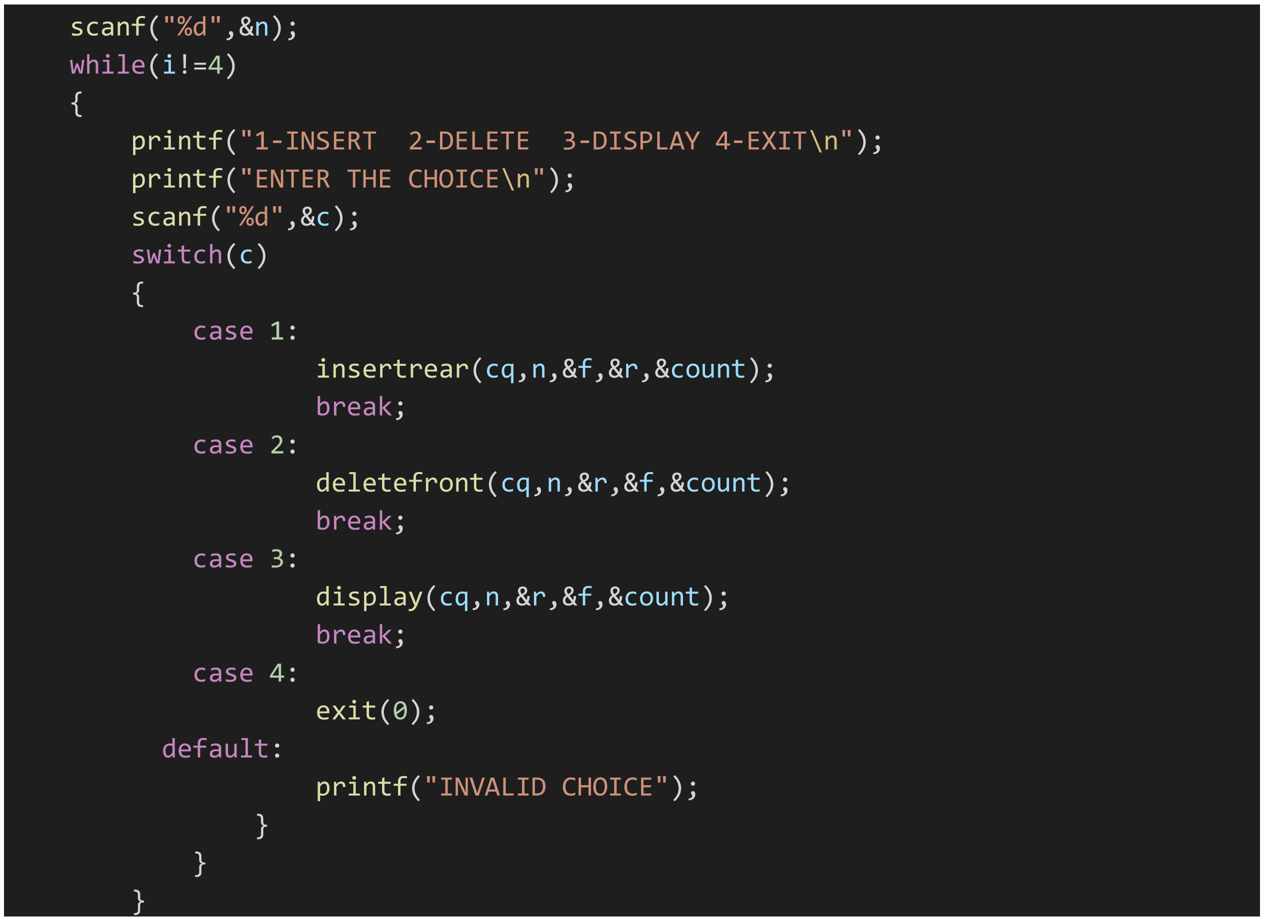
c) Display

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

```c
#include<stdio.h>
#include<conio.h>
#include<process.h>
void insertrear(int cq[10],int n,int *f,int *r,int *count)
{
    if(*count==n)
    {
        printf("CIRCULAR QUEUE OVERFLOW\n");
        return;
    }
    int item;
    printf("ENTER THE ITEM TO BE INSERTED\n");
    scanf("%d",&item);
```

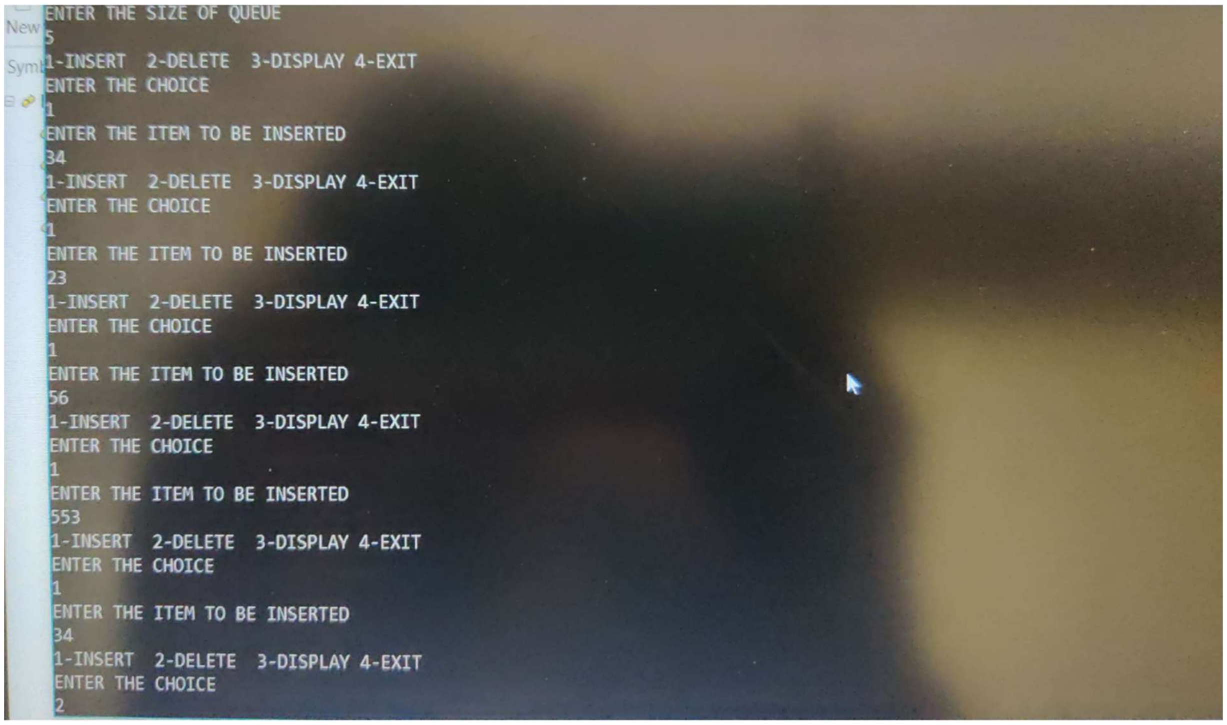


```
    *r=(*r+1)%n;
    cq[*r]=item;
    (*count)++;
}
void deletefront(int cq[10],int n,int *r,int *f,int *count)
{
    if(*count==0)
    {
        printf("THE CIRCULAR QUEUE IS EMPTY\n");
    }
    else
    {
    printf("THE ITEM DELETED IS =%d\n",cq[*f]);
    *f=(*f+1)%n;
}
      (*count)--;
}
void display(int cq[10],int n,int *r,int *f,int *count)
{
    int i;
    if(*count==0)
    {
        printf("THE CIRCULAR QUEUE IS EMPTY\n");
        return;
    }
    else
    {
    printf("THE CONTENTS OF THE CIRCULAR QUEUE IS=  ");
    for(i=1;i<=*count;i++)
    {
        printf("%d\t",cq[*f]);
        *f=(*f+1)%n;
    }
    printf("\n");
}
}
int main()
{
    int c,i;
    int count=0;
    int r=-1;
    int f=0;
    int n;
    int cq[10];
    printf("ENTER THE SIZE OF CIRCULAR QUEUE\n");
```

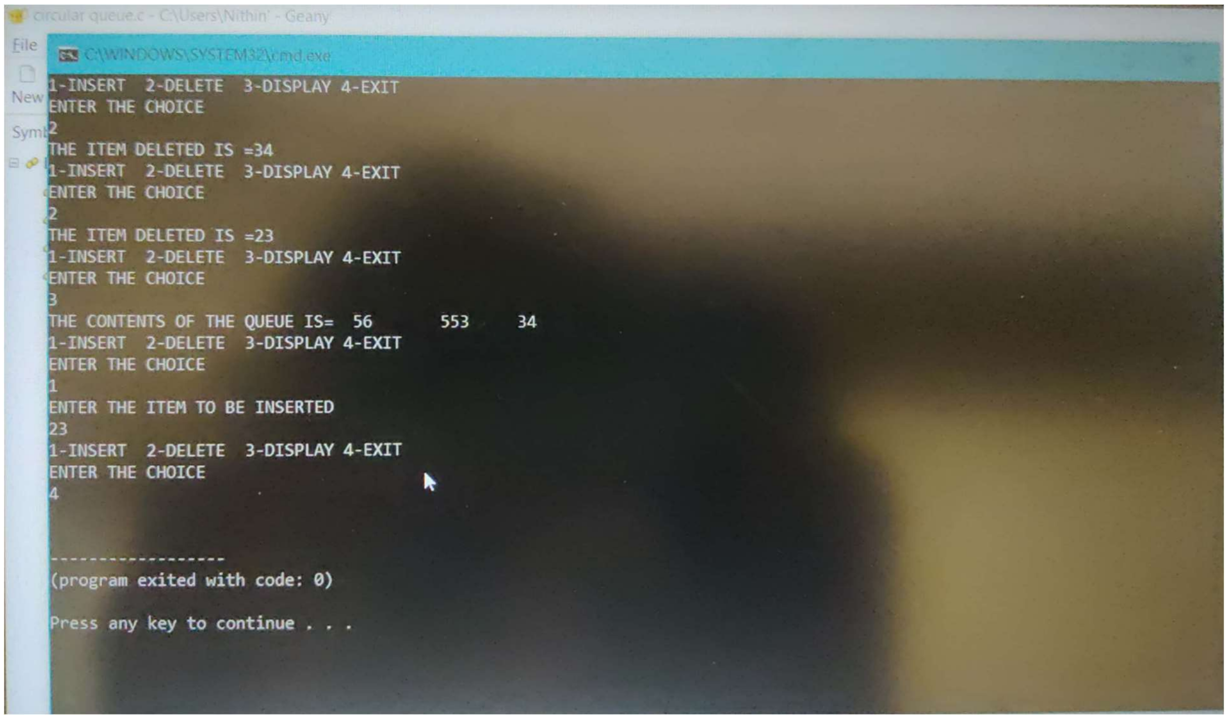


```
    scanf("%d",&n);
    while(i!=4)
    {
        printf("1-INSERT  2-DELETE  3-DISPLAY 4-EXIT\n");
        printf("ENTER THE CHOICE\n");
        scanf("%d",&c);
        switch(c)
        {
            case 1:
                    insertrear(cq,n,&f,&r,&count);
                    break;
            case 2:
                    deletefront(cq,n,&r,&f,&count);
                    break;
            case 3:
                    display(cq,n,&r,&f,&count);
                    break;
            case 4:
                    exit(0);
          default:
                    printf("INVALID CHOICE");
                }
            }
        }
```

Output:



```
ENTER THE SIZE OF QUEUE
5
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
1
ENTER THE ITEM TO BE INSERTED
34
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
1
ENTER THE ITEM TO BE INSERTED
23
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
1
ENTER THE ITEM TO BE INSERTED
56
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
1
ENTER THE ITEM TO BE INSERTED
553
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
1
ENTER THE ITEM TO BE INSERTED
34
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
2
```



```
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
2
THE ITEM DELETED IS =34
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
2
THE ITEM DELETED IS =23
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
3
THE CONTENTS OF THE QUEUE IS=  56      553     34
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
1
ENTER THE ITEM TO BE INSERTED
23
1-INSERT  2-DELETE  3-DISPLAY 4-EXIT
ENTER THE CHOICE
4


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

Press any key to continue . . .
```

Lab Program 5:

WAP to Implement Singly Linked List with following operations

a) Create a linked list.

b) Insertion of a node at first position, at any position and at end of list.

c) Display the contents of the linked list.

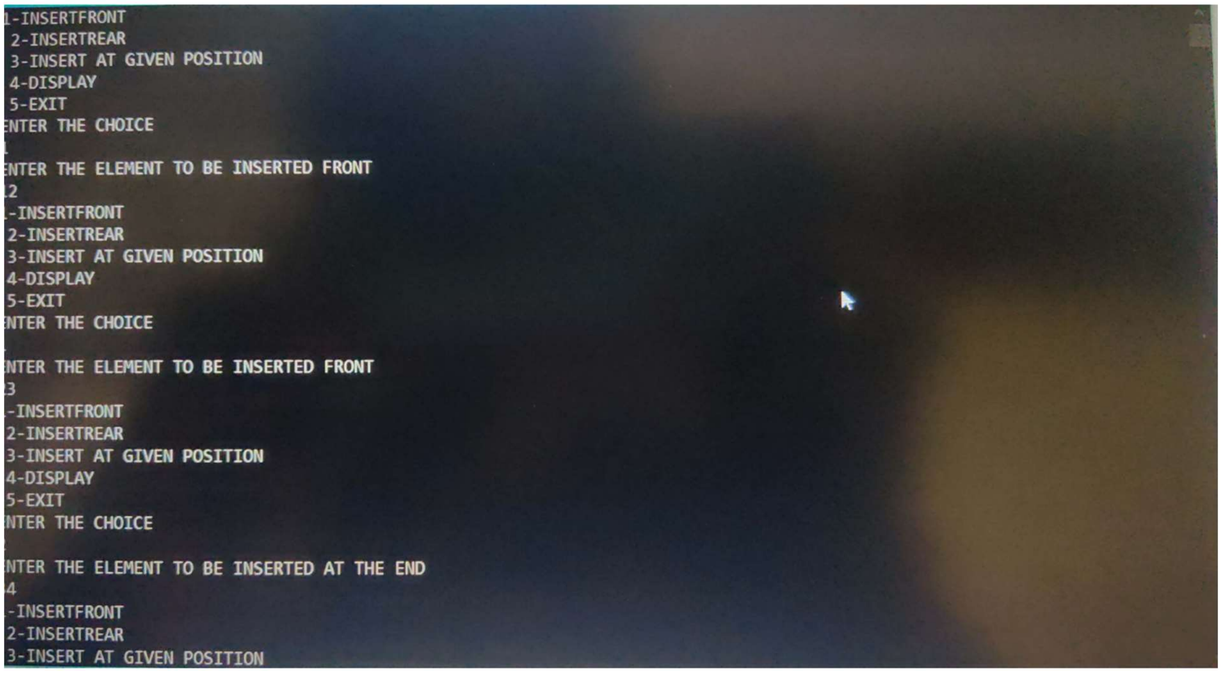
```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<stdlib.h>
struct node{
    int inf;
    struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
```

```c
        printf("MEMORY IS FULL\n");
        exit(0);
    }
    else
    return x;
}
NODE insertfront(NODE first,int item)
{
    NODE temp;
    temp=getnode();
    temp->inf=item;
    temp->link=NULL;
    if(first==NULL)
    {
        return temp;
    }
    temp->link=first;
    first=temp;
    return first;
}
NODE insertrear(NODE first,int item)
{
    NODE temp,cur;
    temp=getnode();
    temp->inf=item;
    temp->link=NULL;
    if(first==NULL)
    {
        return temp;
    }
    cur=first;
    while(cur->link != NULL)
    {
        cur=cur->link;
    }
    cur->link=temp;
    return first;
}

NODE insertpos(int item,int pos,NODE first)
{
NODE temp,cur,prev;
int count;
temp=getnode();
temp->inf=item;
```

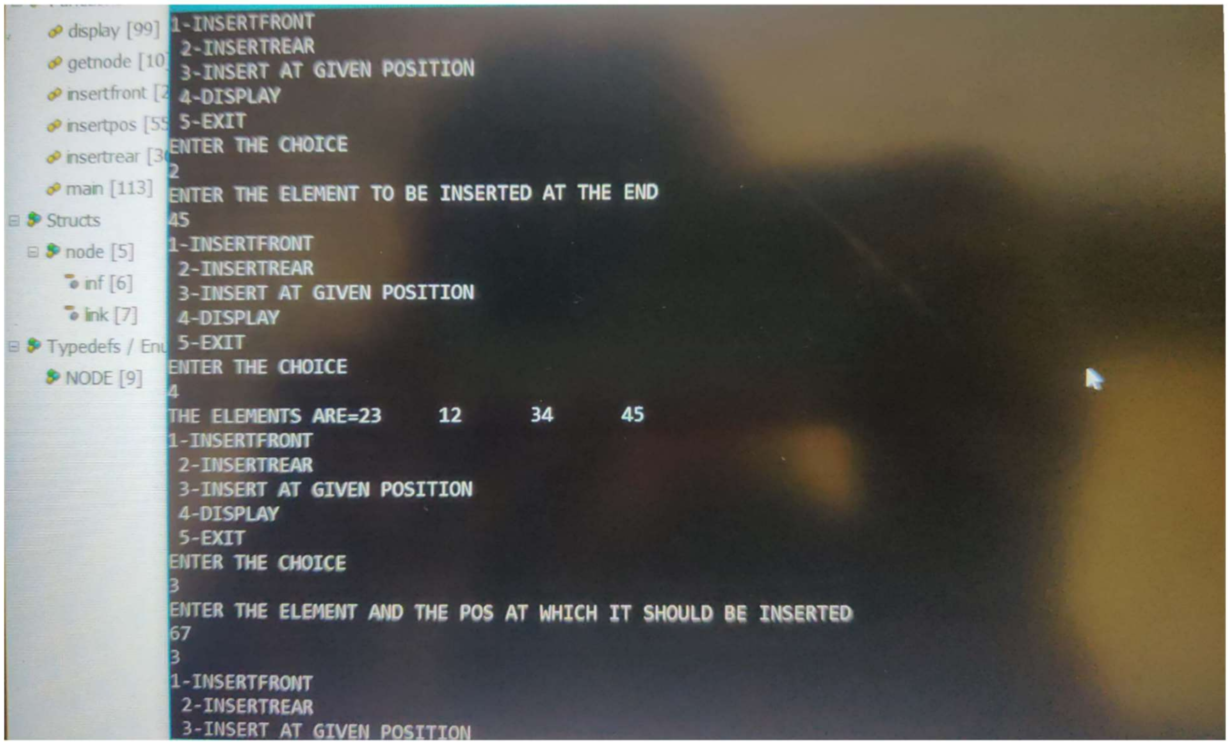
```c
temp->link=NULL;
if(first==NULL&&pos==1)
{
return temp;
}
if(first==NULL)
{
printf("invalid position\n");
return first;
}
if(pos==1)
{
temp->link=first;
first=temp;
return temp;
}
count=1;
prev=NULL;
cur=first;
while(cur!=NULL&&count!=pos)
{
prev=cur;
cur=cur->link;
count++;
}
if(count==pos)
{

prev->link=temp;
temp->link=cur;
return first;
}
printf("invalid position\n");
return first;
}



void display(NODE first)
{
    NODE temp;
    if(first==NULL){
        printf("THE LIST IS EMPTY\n");
    }
    printf("THE ELEMENTS ARE=");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\t",temp->inf);
    }
    printf("\n");
}

int main()
{
    int c,item,pos;
    NODE first=NULL;
    for(;;)
    {
        printf("1-INSERTFRONT \n 2-INSERTREAR \n 3-INSERT AT GIVEN POSITION \n 4-
DISPLAY \n 5-EXIT\n");
        printf("ENTER THE CHOICE\n");
        scanf("%d",&c);
        switch(c)
        {
            case 1:
            printf("ENTER THE ELEMENT TO BE INSERTED FRONT\n");
            scanf("%d",&item);
            first=insertfront(first,item);
            break;
            case 2:
            printf("ENTER THE ELEMENT TO BE INSERTED AT THE END\n");
            scanf("%d",&item);
            first=insertrear(first,item);
            break;
              case 3:
              printf("ENTER THE ELEMENT AND THE POS AT WHICH IT SHOULD BE INSERTE
D\n");
            scanf("%d",&item);
            scanf("%d",&pos);
            first=insertpos(item,pos,first);
            break;
            case 4:
            display(first);
            break;
            case 5:
            exit(0);
         default:
         printf("INVALID CHOICE\n");
     }
 }
```

```
}
```

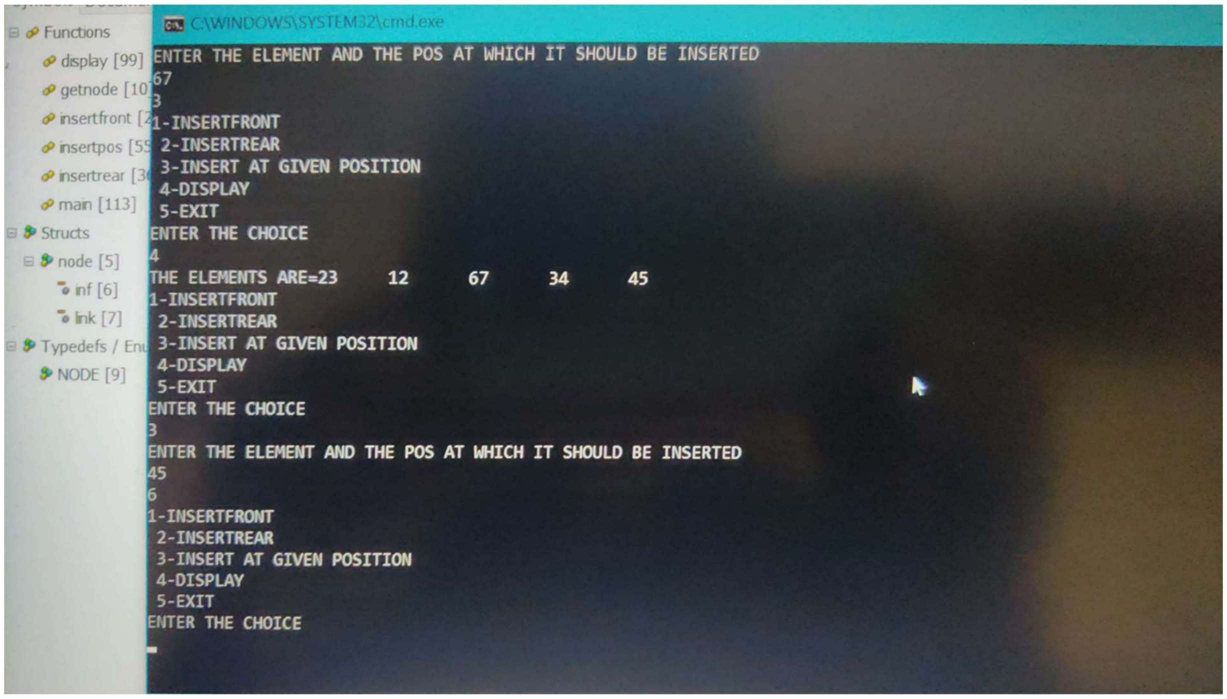
Output:



```
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DISPLAY
 5-EXIT
NTER THE CHOICE
1
NTER THE ELEMENT TO BE INSERTED FRONT
12
-INSERTFRONT
2-INSERTREAR
3-INSERT AT GIVEN POSITION
4-DISPLAY
5-EXIT
NTER THE CHOICE

NTER THE ELEMENT TO BE INSERTED FRONT
3
-INSERTFRONT
2-INSERTREAR
3-INSERT AT GIVEN POSITION
4-DISPLAY
5-EXIT
NTER THE CHOICE

NTER THE ELEMENT TO BE INSERTED AT THE END
4
-INSERTFRONT
2-INSERTREAR
3-INSERT AT GIVEN POSITION
```



```
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DISPLAY
 5-EXIT
ENTER THE CHOICE
2
ENTER THE ELEMENT TO BE INSERTED AT THE END
45
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DISPLAY
 5-EXIT
ENTER THE CHOICE
4
THE ELEMENTS ARE=23      12      34      45
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DISPLAY
 5-EXIT
ENTER THE CHOICE
3
ENTER THE ELEMENT AND THE POS AT WHICH IT SHOULD BE INSERTED
67
3
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
```



```
ENTER THE ELEMENT AND THE POS AT WHICH IT SHOULD BE INSERTED
67
3
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DISPLAY
 5-EXIT
ENTER THE CHOICE
4
THE ELEMENTS ARE=23      12      67      34      45
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DISPLAY
 5-EXIT
ENTER THE CHOICE
3
ENTER THE ELEMENT AND THE POS AT WHICH IT SHOULD BE INSERTED
45
6
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DISPLAY
 5-EXIT
ENTER THE CHOICE
```

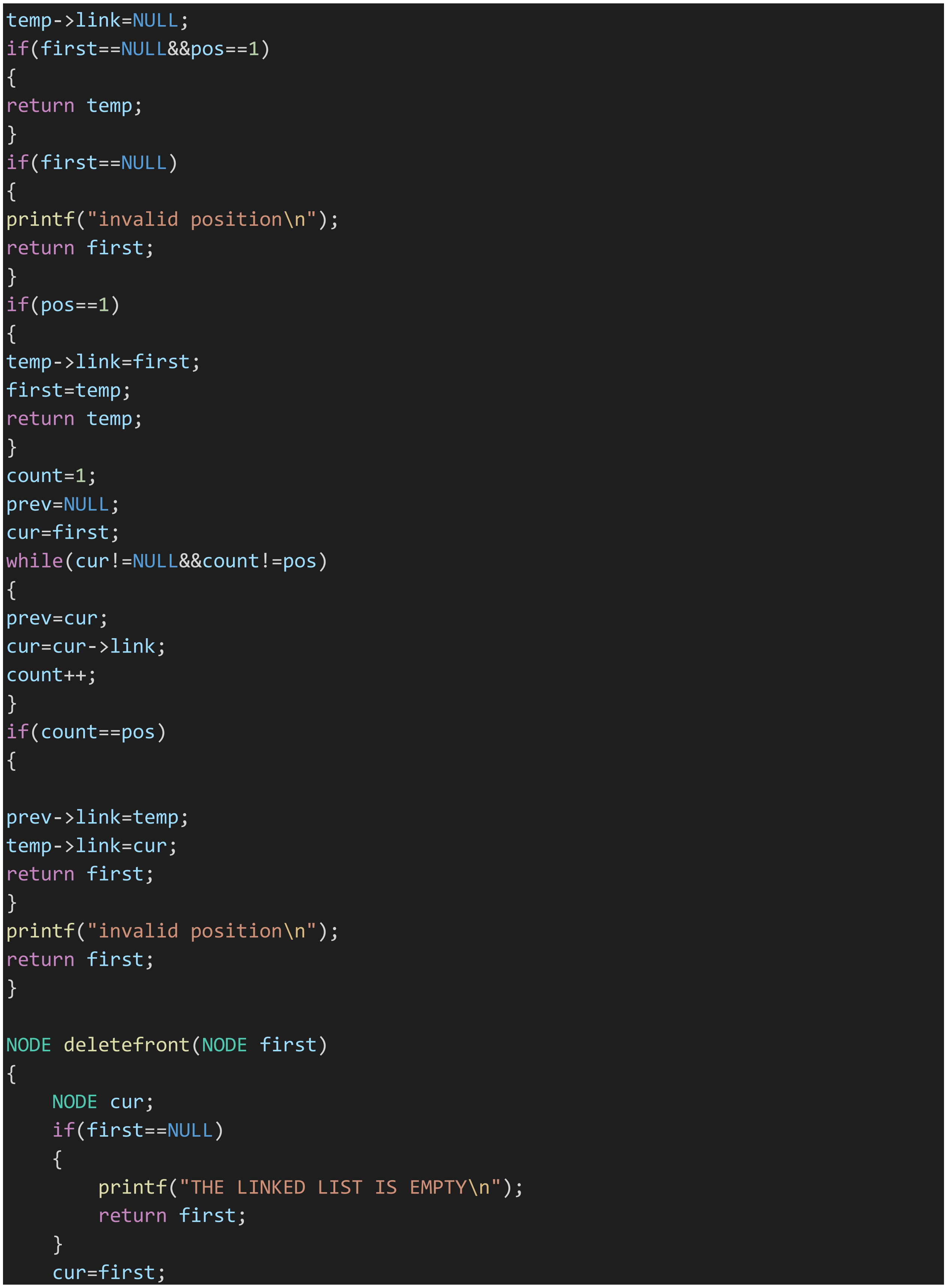
Lab Program 6:

WAP to Implement Singly Linked List with following operations

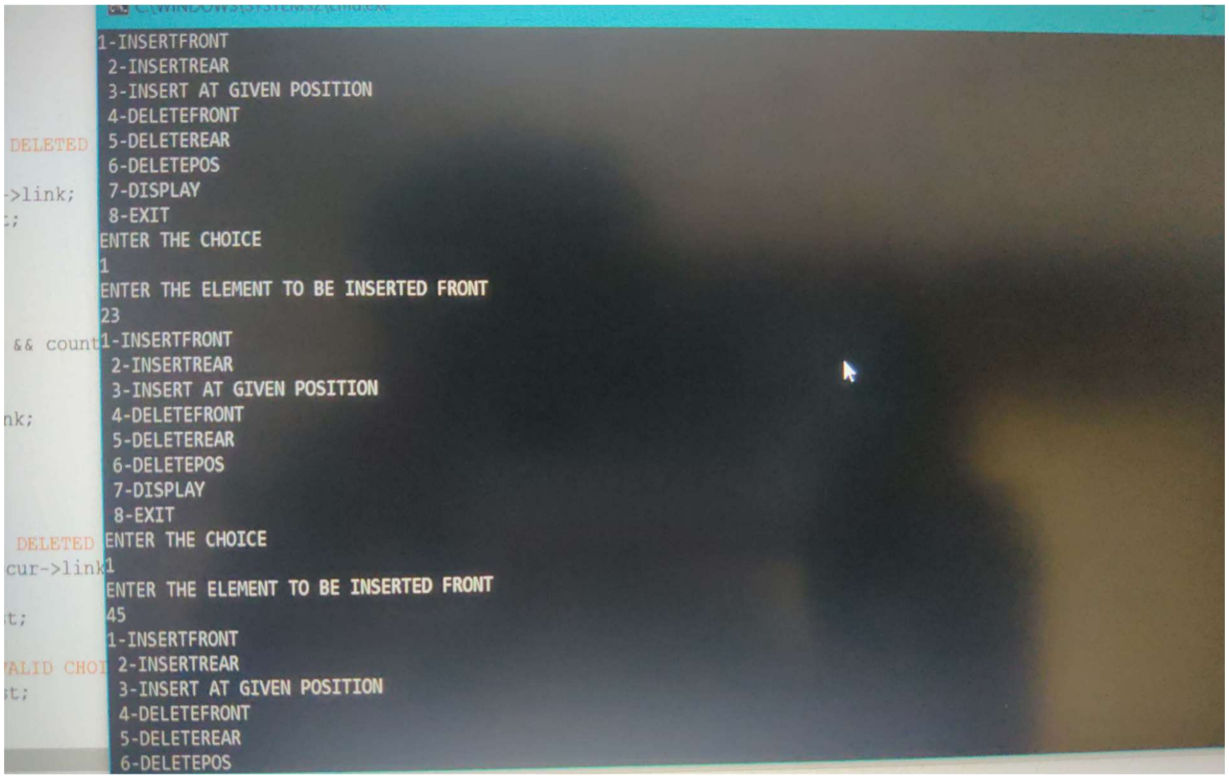
a) Create a linked list.

b) Deletion of first element, specified element and last element in the list.

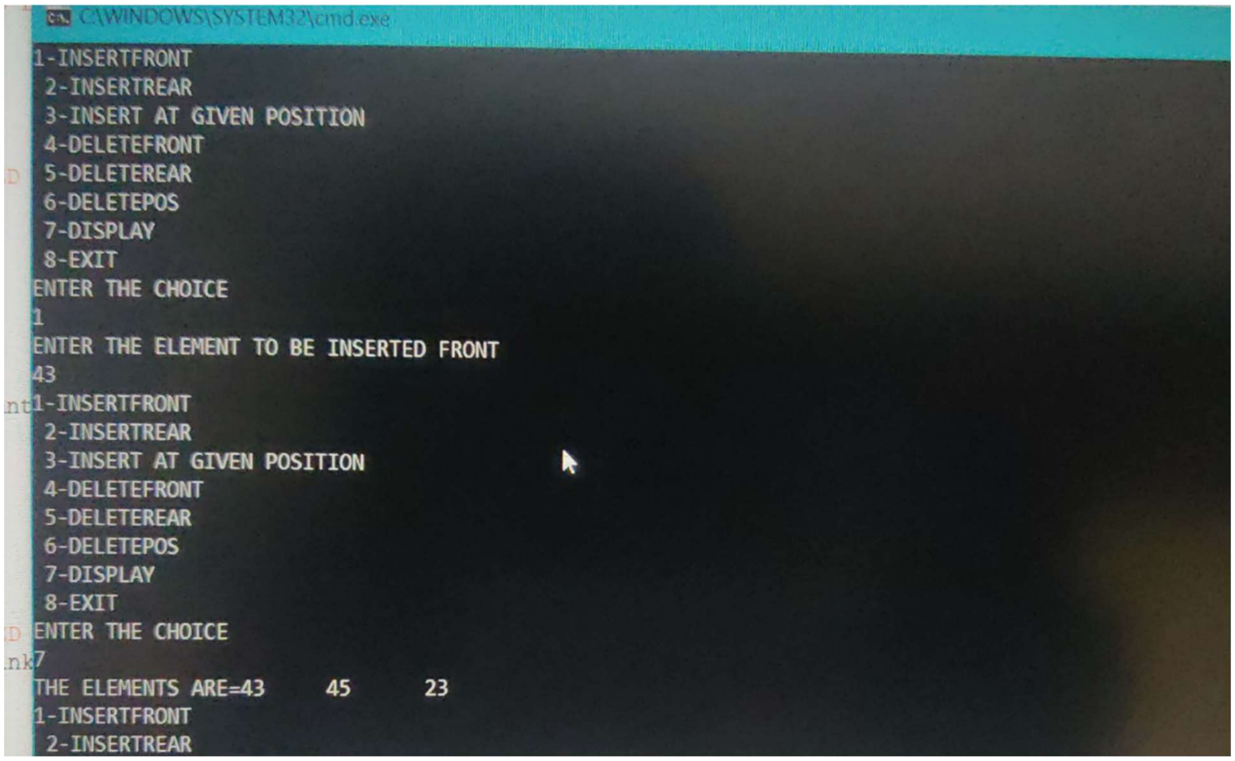
```c
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<stdlib.h>
struct node{
    int info;
    struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
        printf("MEMORY IS FULL\n");
        exit(0);
    }
    else
    return x;
}

NODE insertfront(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 insertrear(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 insertpos(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 deletefront(NODE first)
{
    NODE cur;
    if(first==NULL)
    {
        printf("THE LINKED LIST IS EMPTY\n");
        return first;
    }
    cur=first;
    cur=cur->link;
    printf("THE DELETED ITEM FROM FRONT IS=%d\n",first->info);
    free(first);
    return cur;
}
NODE deleterear(NODE first)
{
    NODE prev,cur;
    if(first==NULL)
    {
        printf("THE LINKED LIST IS EMPTY\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("THE DELETED ITEM FROM REAR IS=%d\n",cur->info);
    free(cur);
    prev->link=NULL;
    return first;
}
NODE deletepos(NODE first,int pos)
{
    NODE prev,cur;
    NODE temp;
    int count;
    if(first==NULL)
    {
        printf("THE LINKED LIST IS EMPTY\n");
        return NULL;
    }
    if(pos==1)
    {
        temp=first;
        printf("THE DELETED ITEM FROM POS 1 IS=%d\n",temp->info);
        free(temp);
        first=first->link;
        return first;
    }
    count=1;
    prev=NULL;
    cur=first;
    while(cur!=NULL && count!=pos)
    {
        prev=cur;
        cur=cur->link;
        count++;
    }
    if(count==pos)
    {
        printf("THE DELETED ITEM AT POSITION %d=%d\n",pos,cur->info);
        prev->link=cur->link;
        free(cur);
        return first;
    }
        printf("INVALID CHOICE\n");
        return first;
}

void display(NODE first)
{
    NODE temp;
    if(first==NULL){
        printf("THE LIST IS EMPTY\n");
    }
    printf("THE ELEMENTS ARE=");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\t",temp->info);
    }
    printf("\n");
}


int main()
{
    int c,item,pos;
    NODE first=NULL;
    for(;;)
    {
        printf("1-INSERTFRONT \n 2-INSERTREAR \n 3-INSERT AT GIVEN POSITION \n 4-
DELETEFRONT \n 5-DELETEREAR \n 6-DELETEPOS \n 7-DISPLAY \n 8-EXIT\n");
        printf("ENTER THE CHOICE\n");
        scanf("%d",&c);
        switch(c)
        {
            case 1:
            printf("ENTER THE ELEMENT TO BE INSERTED FRONT\n");
            scanf("%d",&item);
            first=insertfront(first,item);
            break;
            case 2:
            printf("ENTER THE ELEMENT TO BE INSERTED AT THE END\n");
            scanf("%d",&item);
            first=insertrear(first,item);
            break;
              case 3:
              printf("ENTER THE ELEMENT AND THE POS AT WHICH IT SHOULD BE INSERTE
D\n");
            scanf("%d",&item);
            scanf("%d",&pos);
            first=insertpos(item,pos,first);
            break;
            case 4:
            first=deletefront(first);
            break;
            case 5:
            first=deleterear(first);
            break;
            case 6:
            printf("ENTER  THE POS AT WHICH ELEMENT SHOULD BE DELETED\n");
            scanf("%d",&pos);
            first=deletepos(first,pos);
            break;
            case 7:
            display(first);
            break;
            case 8:
            exit(0);
         default:
         printf("INVALID CHOICE\n");
     }
 }
}
```

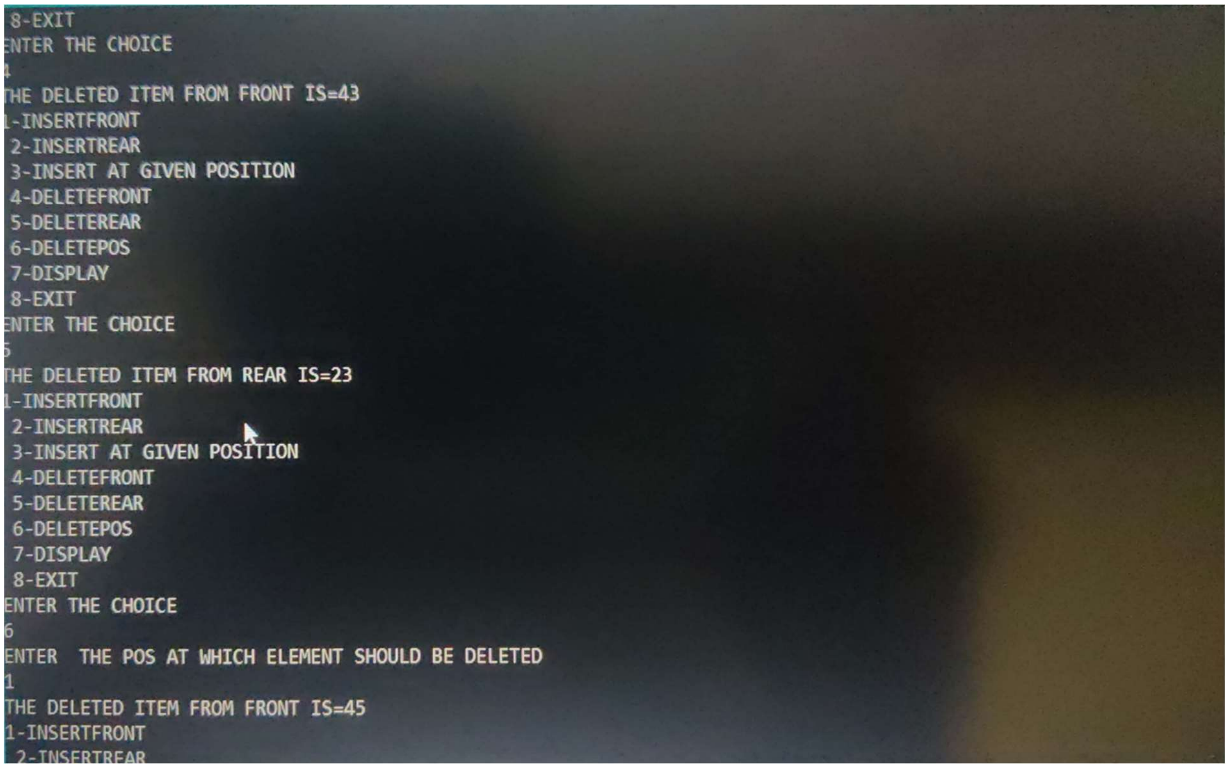
Output:



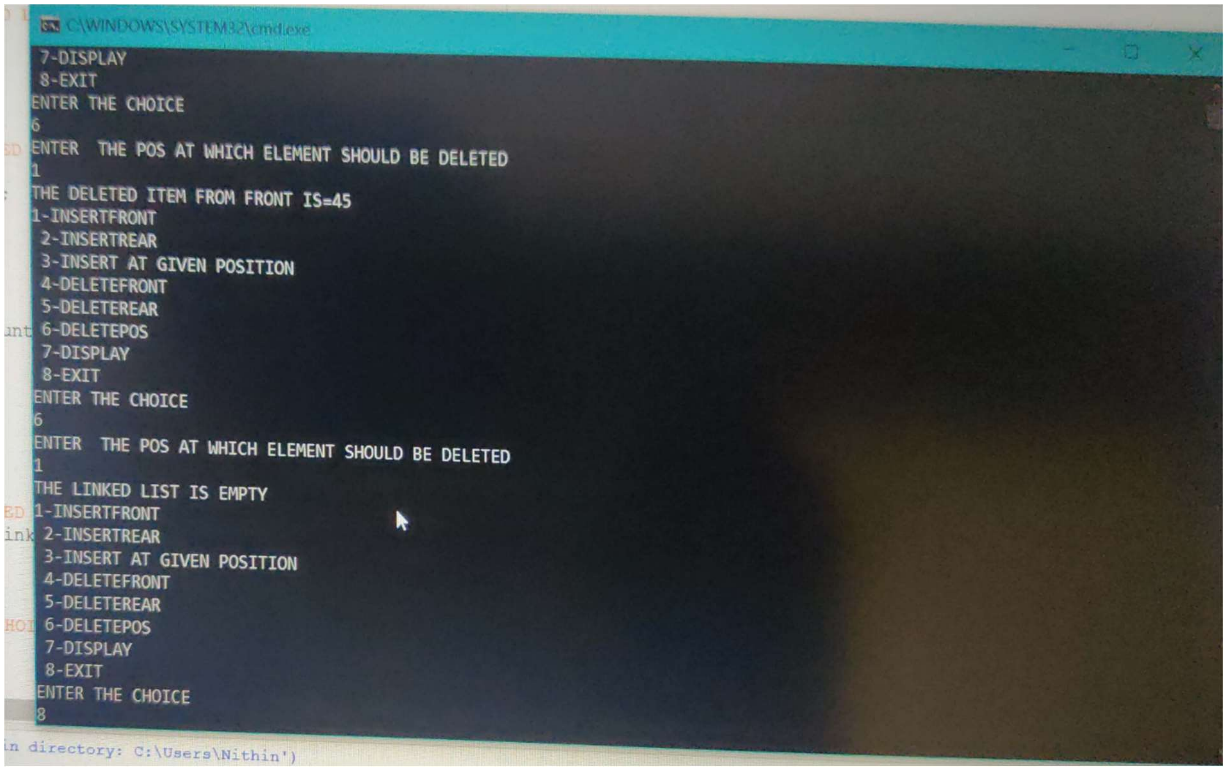
```
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DELETEFRONT
 5-DELETEREAR
 6-DELETEPOS
 7-DISPLAY
 8-EXIT
ENTER THE CHOICE
1
ENTER THE ELEMENT TO BE INSERTED FRONT
23
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DELETEFRONT
 5-DELETEREAR
 6-DELETEPOS
 7-DISPLAY
 8-EXIT
ENTER THE CHOICE
1
ENTER THE ELEMENT TO BE INSERTED FRONT
45
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DELETEFRONT
 5-DELETEREAR
 6-DELETEPOS
```



```
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DELETEFRONT
 5-DELETEREAR
 6-DELETEPOS
 7-DISPLAY
 8-EXIT
ENTER THE CHOICE
1
ENTER THE ELEMENT TO BE INSERTED FRONT
43
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DELETEFRONT
 5-DELETEREAR
 6-DELETEPOS
 7-DISPLAY
 8-EXIT
ENTER THE CHOICE
7
THE ELEMENTS ARE=43      45      23
1-INSERTFRONT
 2-INSERTREAR
```



```
 8-EXIT
ENTER THE CHOICE
4
THE DELETED ITEM FROM FRONT IS=43
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DELETEFRONT
 5-DELETEREAR
 6-DELETEPOS
 7-DISPLAY
 8-EXIT
ENTER THE CHOICE
5
THE DELETED ITEM FROM REAR IS=23
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DELETEFRONT
 5-DELETEREAR
 6-DELETEPOS
 7-DISPLAY
 8-EXIT
ENTER THE CHOICE
6
ENTER  THE POS AT WHICH ELEMENT SHOULD BE DELETED
1
THE DELETED ITEM FROM FRONT IS=45
1-INSERTFRONT
 2-INSERTREAR
```



C:\WINDOWS\SYSTEM32\cmd.exe

```
 7-DISPLAY
 8-EXIT
ENTER THE CHOICE
6
ENTER  THE POS AT WHICH ELEMENT SHOULD BE DELETED
1
THE DELETED ITEM FROM FRONT IS=45
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DELETEFRONT
 5-DELETEREAR
 6-DELETEPOS
 7-DISPLAY
 8-EXIT
ENTER THE CHOICE
6
ENTER  THE POS AT WHICH ELEMENT SHOULD BE DELETED
1
THE LINKED LIST IS EMPTY
1-INSERTFRONT
 2-INSERTREAR
 3-INSERT AT GIVEN POSITION
 4-DELETEFRONT
 5-DELETEREAR
 6-DELETEPOS
 7-DISPLAY
 8-EXIT
ENTER THE CHOICE
8
```

```
in directory: C:\Users\Nithin')
```

Lab Program 7:

WAP Implement Single Link List with following operations

a) Sort the linked list.

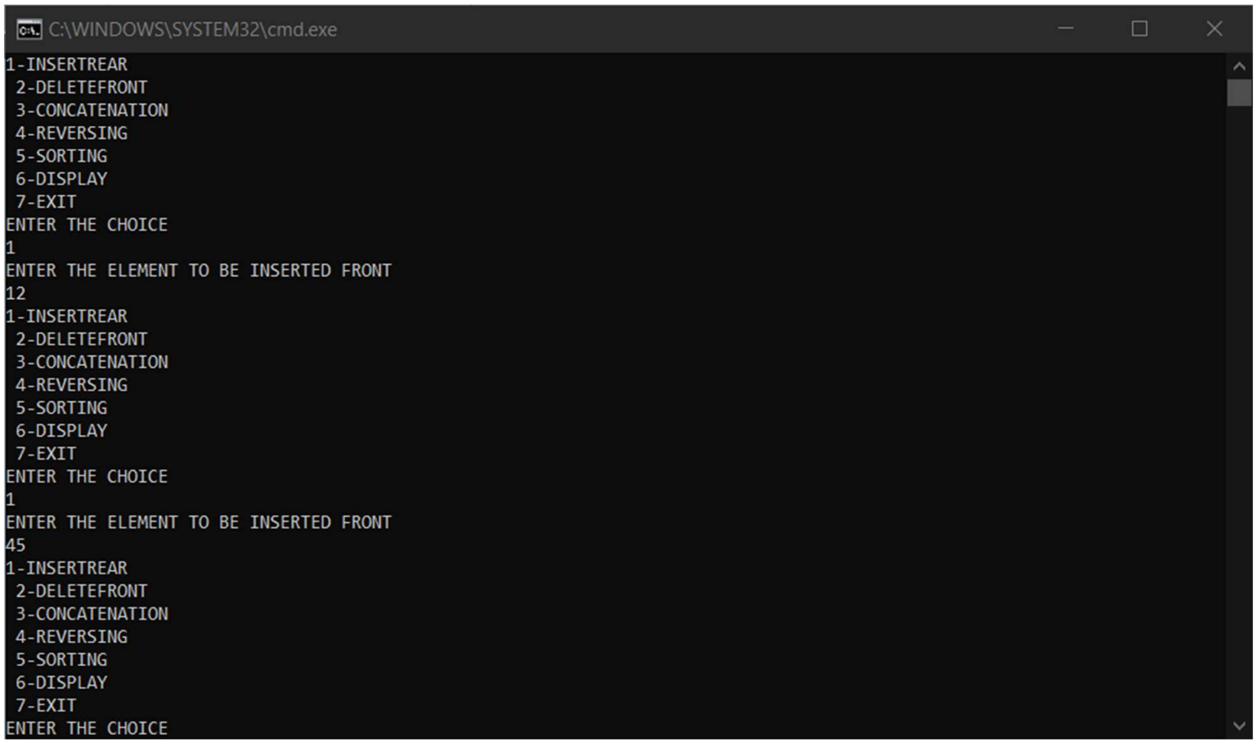
b) Reverse the linked list.

c) Concatenation of two linked lists

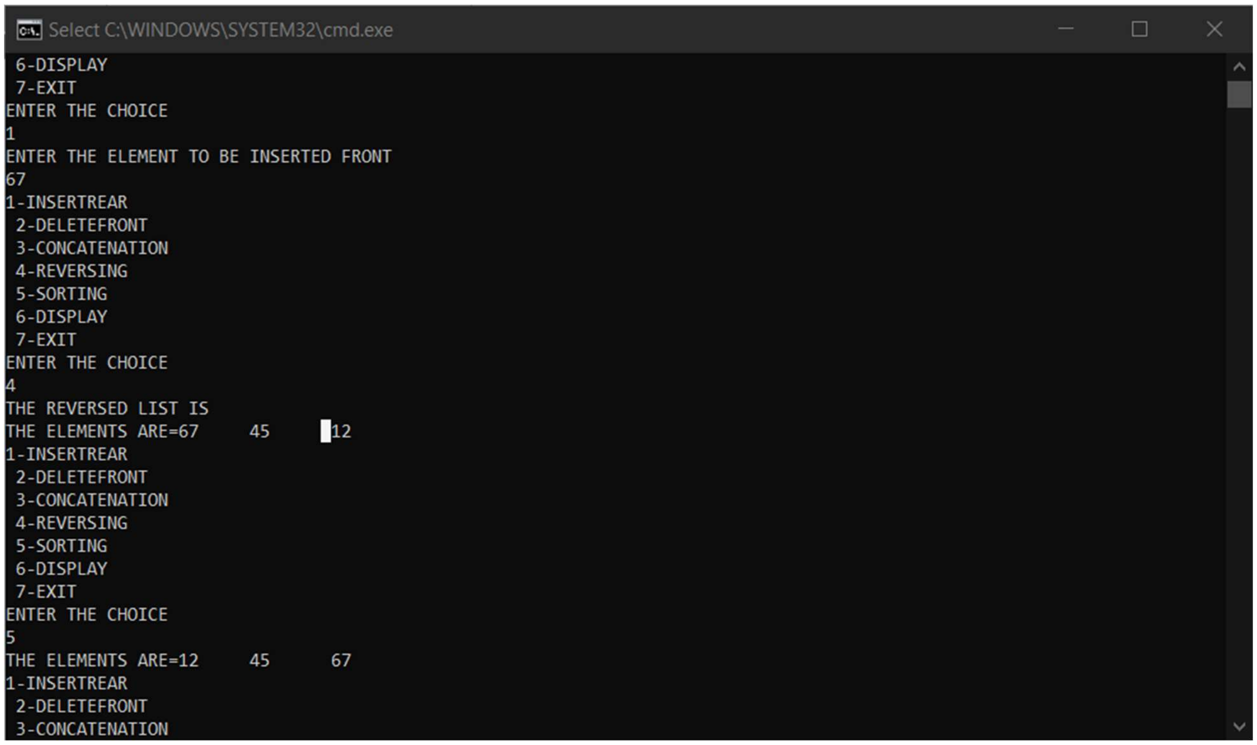
```c
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<stdlib.h>
struct node{
    int info;
    struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if(x==NULL)
    {
        printf("MEMORY IS FULL\n");
        exit(0);
    }
    else
    return x;
}
NODE insertrear(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 deletefront(NODE first)
{
    NODE cur;
    if(first==NULL)
    {
        printf("THE LINKED LIST IS EMPTY\n");
        return first;
    }
    cur=first;
    cur=cur->link;
    printf("THE DELETED ITEM FROM FRONT IS=%d\n",first->info);
    free(first);
    return cur;
}
NODE reverse(NODE first)
{
    NODE cur,temp;
    cur=NULL;
    while(first!=NULL)
    {
        temp=first;
        first=first->link;
        temp->link=cur;
        cur=temp;
    }
    return cur;
}
NODE concat(NODE first,NODE second)
{
    NODE cur;
    if(first==NULL)
    return second;
    if(second==NULL)
    return first;
    cur=first;
    while(cur->link!=NULL)
    {
        cur=cur->link;
    }
    cur->link=second;
    return first;
```

```c
}
void display(NODE first)
{
    NODE temp;
    if(first==NULL){
        printf("THE LIST IS EMPTY\n");
    }
    printf("THE ELEMENTS ARE=");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\t",temp->info);
    }
    printf("\n");
}
NODE sort(NODE first)
{
    int swapped;
    NODE ptr1;
    NODE lptr = NULL;
    if (first == NULL)
        return NULL;
    do
    {
        swapped = 0;
        ptr1 = first;

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

                int tem = ptr1->info;
                ptr1->info = ptr1->link->info;
                ptr1->link->info = tem;
                    swapped = 1;
            }
            ptr1 = ptr1->link;
        }
        lptr = ptr1;
    } while (swapped);
    return first;
}
int main()
{
    int item;
    int n,c;
    NODE first=NULL,fir,sec;
    int i;
    for(;;)
    {
        printf("1-INSERTREAR \n 2-DELETEFRONT \n 3-CONCATENATION  \n 4-
REVERSING \n 5-SORTING \n 6-DISPLAY  \n 7-EXIT\n");
        printf("ENTER THE CHOICE\n");
        scanf("%d",&c);
        switch(c)
        {
            case 1:
            printf("ENTER THE ELEMENT TO BE INSERTED FRONT\n");
            scanf("%d",&item);
            first=insertrear(first,item);
            break;
            case 2:
            first=deletefront(first);
            break;
            case 3:
            printf("ENTER THE NUMBER OF NODES IN FIRST LIST\n");
            scanf("%d",&n);
            fir=NULL;
            for(i=0;i<n;i++)
            {
                printf("ENTER THE ITEM TO BE INSERTED\n");
                scanf("%d",&item);
                fir=insertrear(fir,item);
            }
            printf("ENTER THE NUMBER OF NODES IN SECOND LIST\n");
            scanf("%d",&n);
            sec=NULL;
            for(i=0;i<n;i++)
            {
                printf("ENTER THE ITEM TO BE INSERTED\n");
                scanf("%d",&item);
                sec=insertrear(sec,item);
            }
              fir=concat(fir,sec);
                display(fir);
                break;
            case 4:
            first=reverse(first);
            printf("THE REVERSED LIST IS\n");
            display(first);
            break;
            case 5:
            first=sort(first);
            display(first);
            break;
            case 6:
            display(first);
            break;
            case 7:
            exit(0);
         default:
         printf("INVALID CHOICE\n");
     }
 }
}
```

Output:

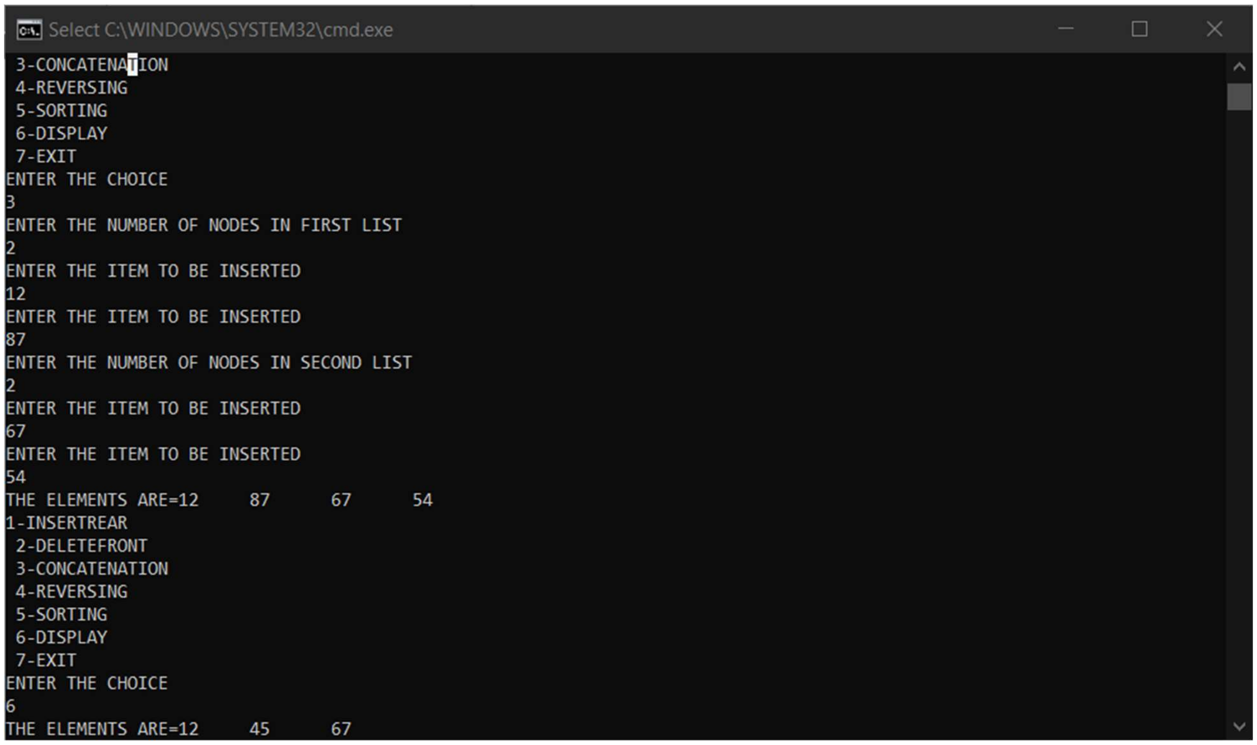


```
C:\WINDOWS\SYSTEM32\cmd.exe
1-INSERTREAR
 2-DELETEFRONT
 3-CONCATENATION
 4-REVERSING
 5-SORTING
 6-DISPLAY
 7-EXIT
ENTER THE CHOICE
1
ENTER THE ELEMENT TO BE INSERTED FRONT
12
1-INSERTREAR
 2-DELETEFRONT
 3-CONCATENATION
 4-REVERSING
 5-SORTING
 6-DISPLAY
 7-EXIT
ENTER THE CHOICE
1
ENTER THE ELEMENT TO BE INSERTED FRONT
45
1-INSERTREAR
 2-DELETEFRONT
 3-CONCATENATION
 4-REVERSING
 5-SORTING
 6-DISPLAY
 7-EXIT
ENTER THE CHOICE
```



Select C:\WINDOWS\SYSTEM32\cmd.exe

```
 6-DISPLAY
 7-EXIT
ENTER THE CHOICE
1
ENTER THE ELEMENT TO BE INSERTED FRONT
67
1-INSERTREAR
 2-DELETEFRONT
 3-CONCATENATION
 4-REVERSING
 5-SORTING
 6-DISPLAY
 7-EXIT
ENTER THE CHOICE
4
THE REVERSED LIST IS
THE ELEMENTS ARE=67     45      12
1-INSERTREAR
 2-DELETEFRONT
 3-CONCATENATION
 4-REVERSING
 5-SORTING
 6-DISPLAY
 7-EXIT
ENTER THE CHOICE
5
THE ELEMENTS ARE=12     45      67
1-INSERTREAR
 2-DELETEFRONT
 3-CONCATENATION
```



Select C:\WINDOWS\SYSTEM32\cmd.exe

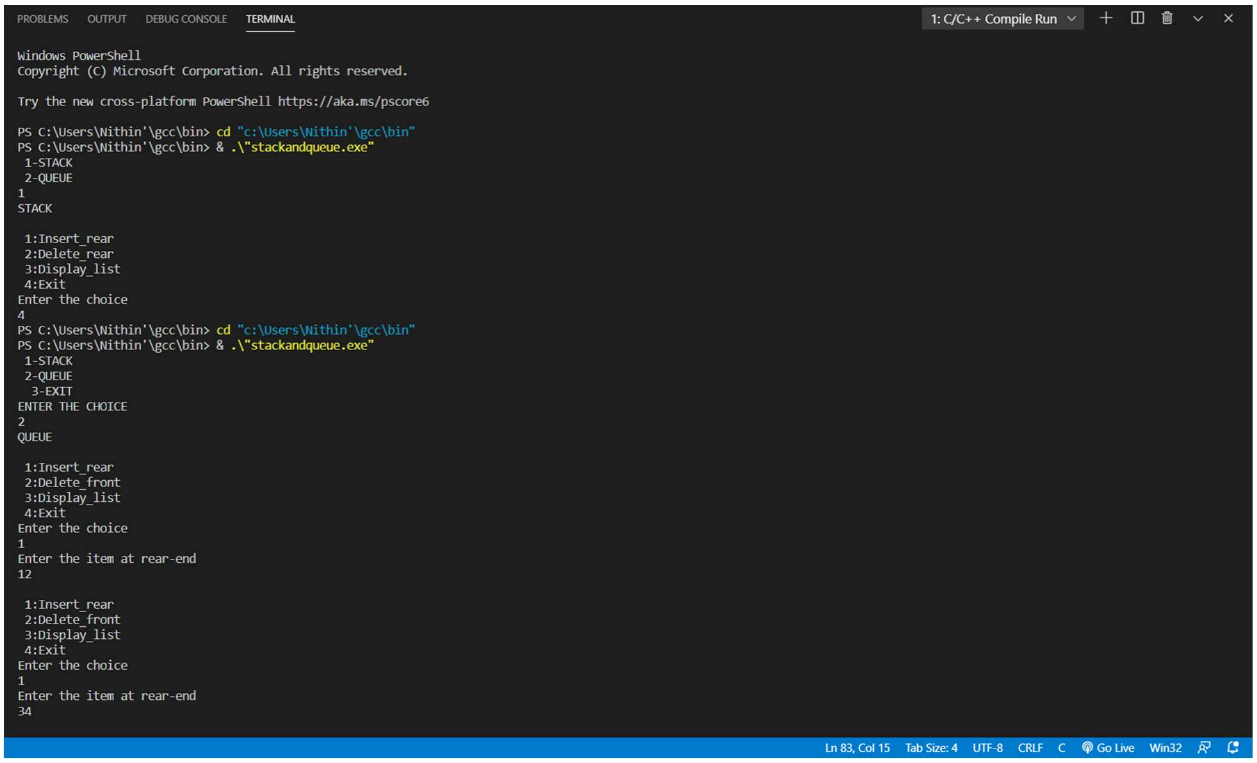
```
 3-CONCATENATION
 4-REVERSING
 5-SORTING
 6-DISPLAY
 7-EXIT
ENTER THE CHOICE
3
ENTER THE NUMBER OF NODES IN FIRST LIST
2
ENTER THE ITEM TO BE INSERTED
12
ENTER THE ITEM TO BE INSERTED
87
ENTER THE NUMBER OF NODES IN SECOND LIST
2
ENTER THE ITEM TO BE INSERTED
67
ENTER THE ITEM TO BE INSERTED
54
THE ELEMENTS ARE=12     87      67      54
1-INSERTREAR
 2-DELETEFRONT
 3-CONCATENATION
 4-REVERSING
 5-SORTING
 6-DISPLAY
 7-EXIT
ENTER THE CHOICE
6
THE ELEMENTS ARE=12     45      67
```

Lab Program 8:

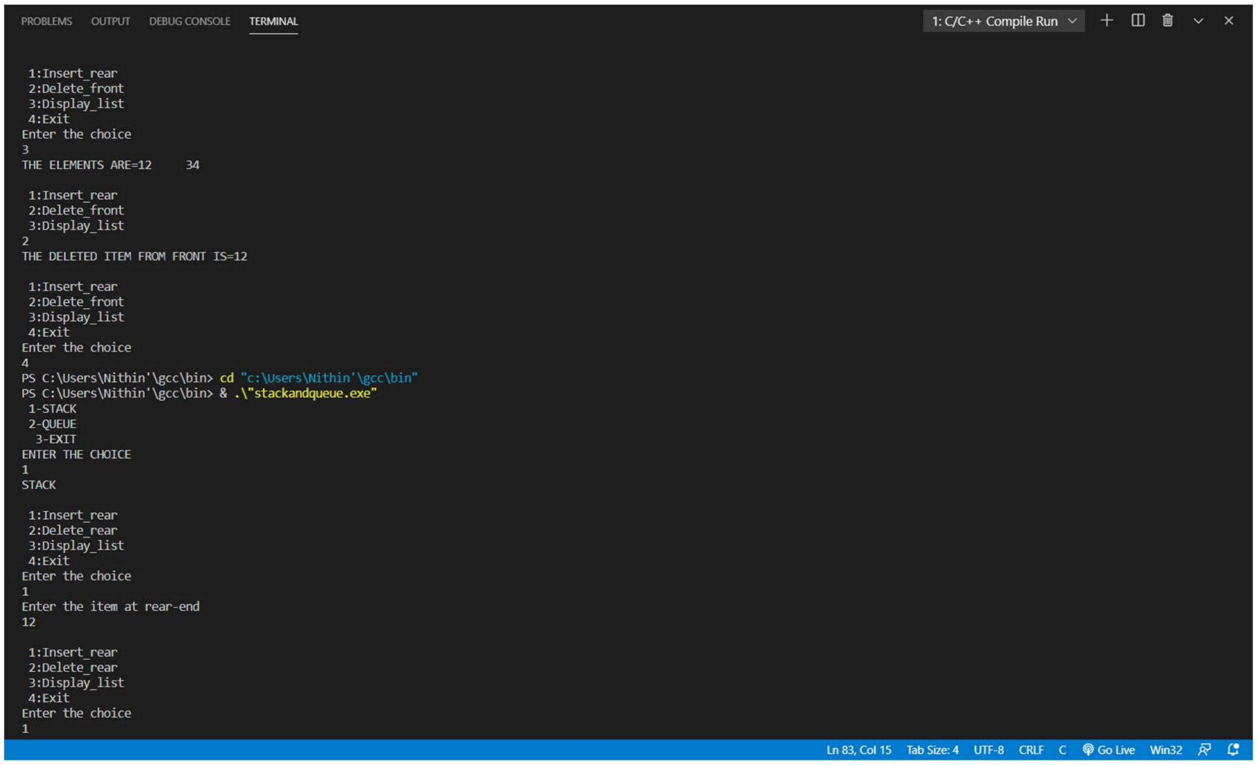
Write a program to implement Stack and Queues using Linked Representation

```c
#include<stdio.h>
#include<conio.h>
#include<process.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 IS FULL\n");
        exit(0);
    }
    else
    return x;
}
NODE insertfront(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 insertrear(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 deletefront(NODE first)
{
    NODE cur;
    if(first==NULL)
    {
        printf("THE  LIST IS EMPTY\n");
        return first;
    }
    cur=first;
    cur=cur->link;
    printf("THE DELETED ITEM FROM FRONT IS=%d\n",first->info);
    free(first);
    return cur;
}
NODE deleterear(NODE first)
{
    NODE prev,cur;
    if(first==NULL)
    {
        printf("THE  LIST IS EMPTY\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("THE DELETED ITEM FROM REAR IS=%d\n",cur->info);
    free(cur);
    prev->link=NULL;
    return first;
}
void display(NODE first)
{
    NODE temp;
    if(first==NULL){
        printf("THE LIST IS EMPTY\n");
    }
    printf("THE ELEMENTS ARE=");
    for(temp=first;temp!=NULL;temp=temp->link)
    {
        printf("%d\t",temp->info);
    }
    printf("\n");
}
int main()
{
    int c,item,pos;
    int n,i;
    int choice;
    NODE first=NULL,sec,fir;
    for(;;)
    {
        printf(" 1-STACK \n 2-QUEUE \n  3-EXIT\n");
        printf("ENTER THE CHOICE\n");
        scanf("%d",&c);
        switch(c)
        {
            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=insertrear(first,item);
            break;
        case 2:first=deleterear(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=insertrear(first,item);
                    break;
            case 2:first=deletefront(first);
                    break;
            case 3:display(first);
                    break;
            default:exit(0);
                    break;
            }
        }
        break;
            case 3:
             exit(0);
         default:
         printf("INVALID CHOICE\n");
```

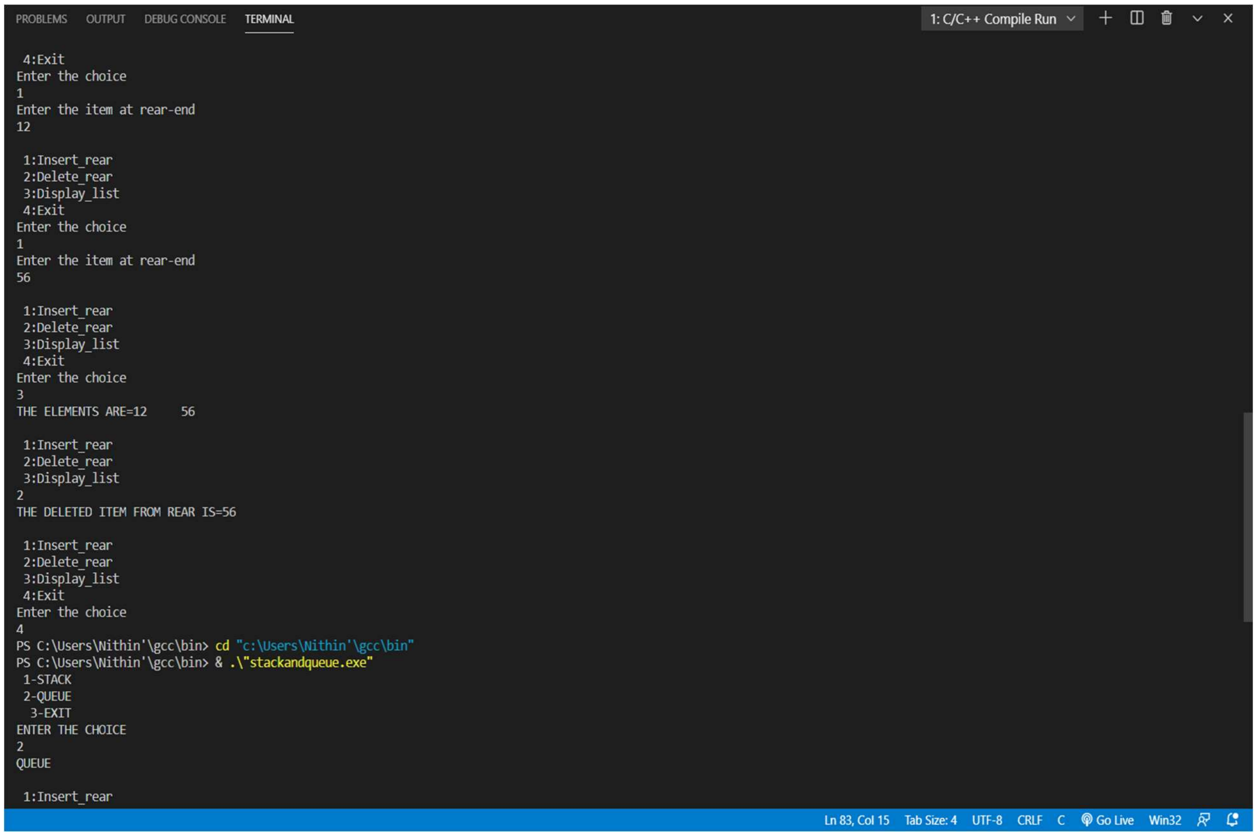
Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Nithin'\gcc\bin> cd "c:\Users\Nithin'\gcc\bin"
PS C:\Users\Nithin'\gcc\bin> & .\"stackandqueue.exe"
 1-STACK
 2-QUEUE
1
STACK

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
4
PS C:\Users\Nithin'\gcc\bin> cd "c:\Users\Nithin'\gcc\bin"
PS C:\Users\Nithin'\gcc\bin> & .\"stackandqueue.exe"
 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
12

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
34
```



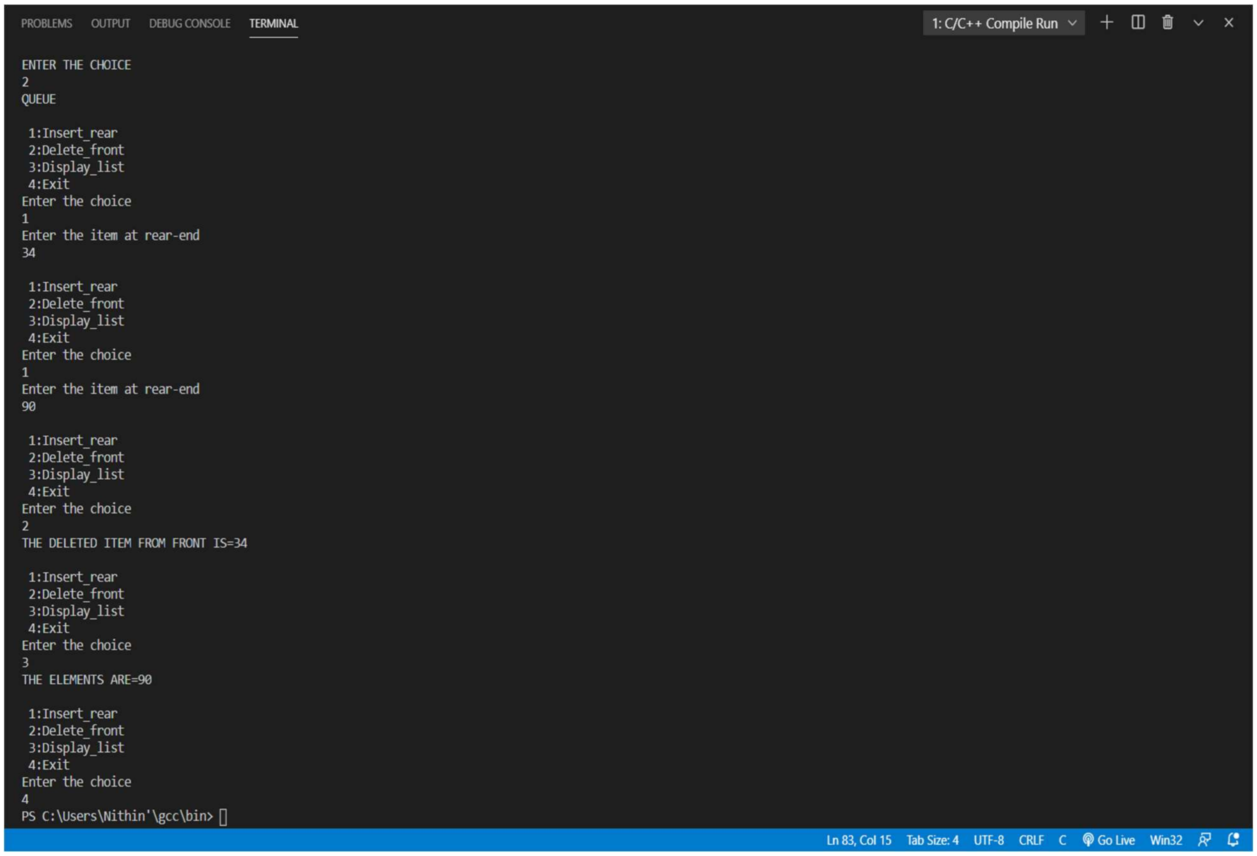
```
 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
3
THE ELEMENTS ARE=12     34

 1:Insert_rear
 2:Delete_front
 3:Display_list
2
THE DELETED ITEM FROM FRONT IS=12

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
4
PS C:\Users\Nithin'\gcc\bin> cd "c:\Users\Nithin'\gcc\bin"
PS C:\Users\Nithin'\gcc\bin> & .\"stackandqueue.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
12

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



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
 4:Exit
Enter the choice
1
Enter the item at rear-end
12

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
56

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
3
THE ELEMENTS ARE=12     56

 1:Insert_rear
 2:Delete_rear
 3:Display_list
2
THE DELETED ITEM FROM REAR IS=56

 1:Insert_rear
 2:Delete_rear
 3:Display_list
 4:Exit
Enter the choice
4
PS C:\Users\Nithin'\gcc\bin> cd "c:\Users\Nithin'\gcc\bin"
PS C:\Users\Nithin'\gcc\bin> & .\"stackandqueue.exe"
 1-STACK
 2-QUEUE
  3-EXIT
ENTER THE CHOICE
2
QUEUE

 1:Insert_rear
```

Ln 83, Col 15 Tab Size: 4 UTF-8 CRLF C Go Live Win32



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
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
34

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
1
Enter the item at rear-end
90

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
2
THE DELETED ITEM FROM FRONT IS=34

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
3
THE ELEMENTS ARE=90

 1:Insert_rear
 2:Delete_front
 3:Display_list
 4:Exit
Enter the choice
4
PS C:\Users\Nithin'\gcc\bin> 
```

Ln 83, Col 15 Tab Size: 4 UTF-8 CRLF C Go Live Win32

Lab Program 9:

WAP Implement doubly link list with primitive operations

a) Create a doubly linked list.

b) Insert a new node to the left of the node.

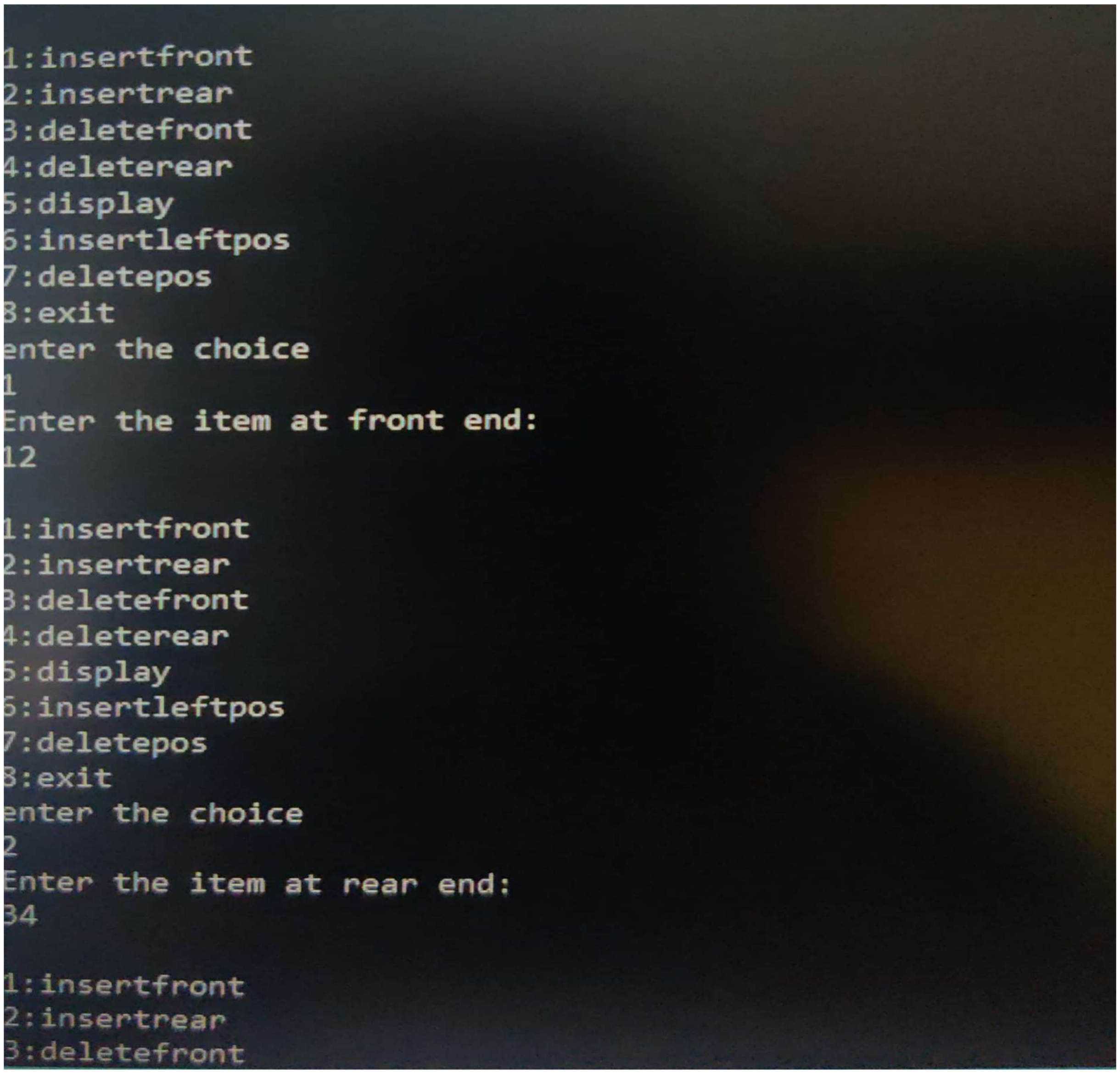
c) Delete the node based on a specific value

d) Display the contents of the list

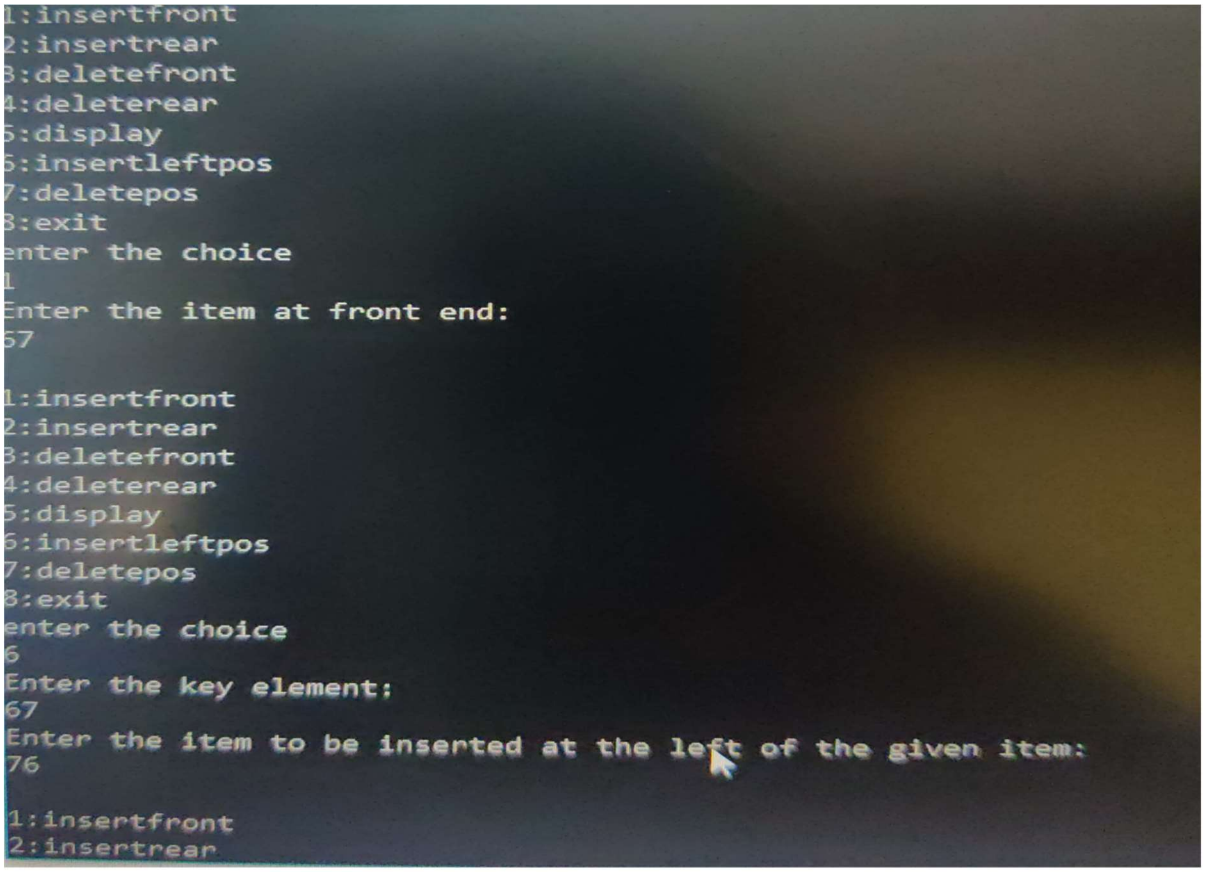
```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <process.h>
struct node
{
    int info;
    struct node *rlink;
    struct node *llink;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(struct node));
    if (x==NULL)
    {
        printf("Memory full\n");
        exit(0);
    }
    return x;
}
NODE insertfront(NODE head,int item)
{
    NODE temp,cur;
    temp=getnode();
    temp->rlink=NULL;
    temp->llink=NULL;
    temp->info=item;
    cur=head->rlink;
    head->rlink=temp;
    temp->llink=head;
    temp->rlink=cur;
    cur->llink=temp;
    return head;
}
NODE insertrear(NODE head,int item)
{
    NODE temp,cur;
    temp=getnode();
    temp->rlink=NULL;
    temp->llink=NULL;
    temp->info=item;
    cur=head->llink;
    head->llink=temp;
    temp->rlink=head;
    cur->rlink=temp;
    temp->llink=cur;
    return head;
}
NODE deletefront(NODE head)
{
    NODE cur,next;
    if(head->rlink==head)
    {
        printf("DOUBLY LINKED LIST IS EMPTY\n");
        return head;
    }
    cur=head->rlink;
    next=cur->rlink;
    head->rlink=next;
    next->llink=head;
    printf("THE ITEM DELETED FROM FRONT=%d\n",cur->info);
    free(cur);
    return head;
}
NODE deleterear(NODE head)
{
    NODE cur,prev;
    if(head->rlink==head)
    {
        printf("DOUBLY LINKED LIST IS EMPTY\n");
        return head;
    }
```

```c
cur=head->llink;
prev=cur->llink;
head->llink=prev;
prev->rlink=head;
printf("THE ITEM DELETED FROM FRONT=%d\n",cur->info);
    free(cur);
    return head;
}
void display(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;
    }
}
NODE insertleftpos(int item,NODE head)
{
    NODE temp,cur,prev;
    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("INVALID ITEM\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 deletepos(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 (item!=cur->info)
        {
            cur=cur->rlink;
        }
        else
        {
                count++;
                prev=cur->llink;
                next=cur->rlink;
                prev->rlink=next;
                next->llink=prev;
                free(cur);
                cur=next;
            }
        }
    if (count==0)
    {
        printf("No such item found in the list\n");
    }
    else
    {
        printf("Removed all the duplicate elements of the given item successfully
\n");
    }
    return head;
}
int main()
{
NODE head;
int item, choice,key;
head=getnode();
head->llink=head;
head->rlink=head;
for(;;)
{
    printf("\n1:insertfront\n2:insertrear\n3:deletefront\n4:deleterear\n5:display
\n6:insertleftpos\n7:deletepos\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);
                head=insertfront(head,item);
                break;
        case 2: printf("Enter the item at rear end:\n");
                scanf("%d",&item);
                head=insertrear(head,item);
                break;
        case 3:head=deletefront(head);
               break;
        case 4:head=deleterear(head);
               break;
        case 5:display(head);
               break;
        case 6:printf("Enter the key element:\n");
               scanf("%d",&key);
               head=insertleftpos(key,head);
               break;
        case 7:printf("Enter the key element whose duplicates should be removed:\
n");
               scanf("%d",&key);
               head=deletepos(key,head);
               break;
        case 8:exit(0);
        default:printf("INVALID CHOICE\n");
        }
    }
}
```

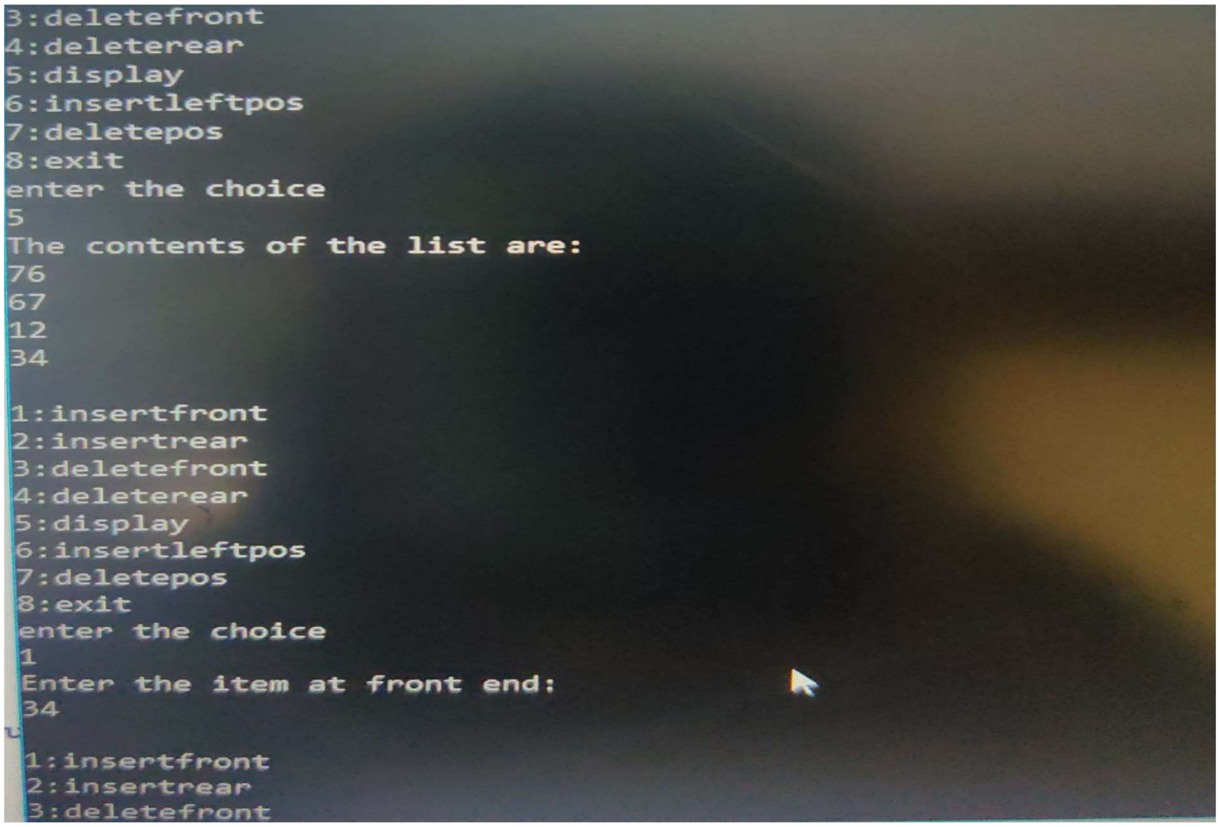
Output:



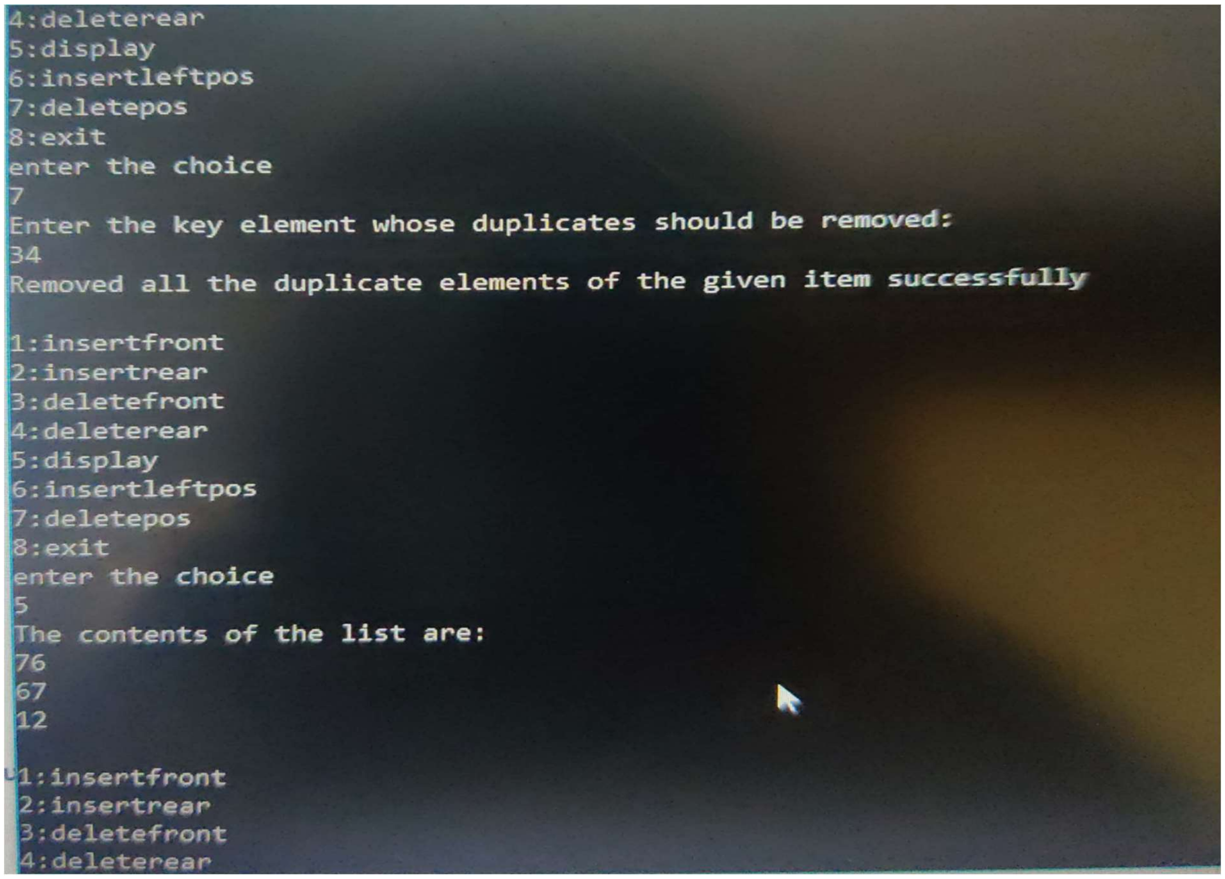
```
1:insertfront
2:insertrear
3:deletefront
4:deleterear
5:display
6:insertleftpos
7:deletepos
8:exit
enter the choice
1
Enter the item at front end:
12

1:insertfront
2:insertrear
3:deletefront
4:deleterear
5:display
6:insertleftpos
7:deletepos
8:exit
enter the choice
2
Enter the item at rear end:
34

1:insertfront
2:insertrear
3:deletefront
```



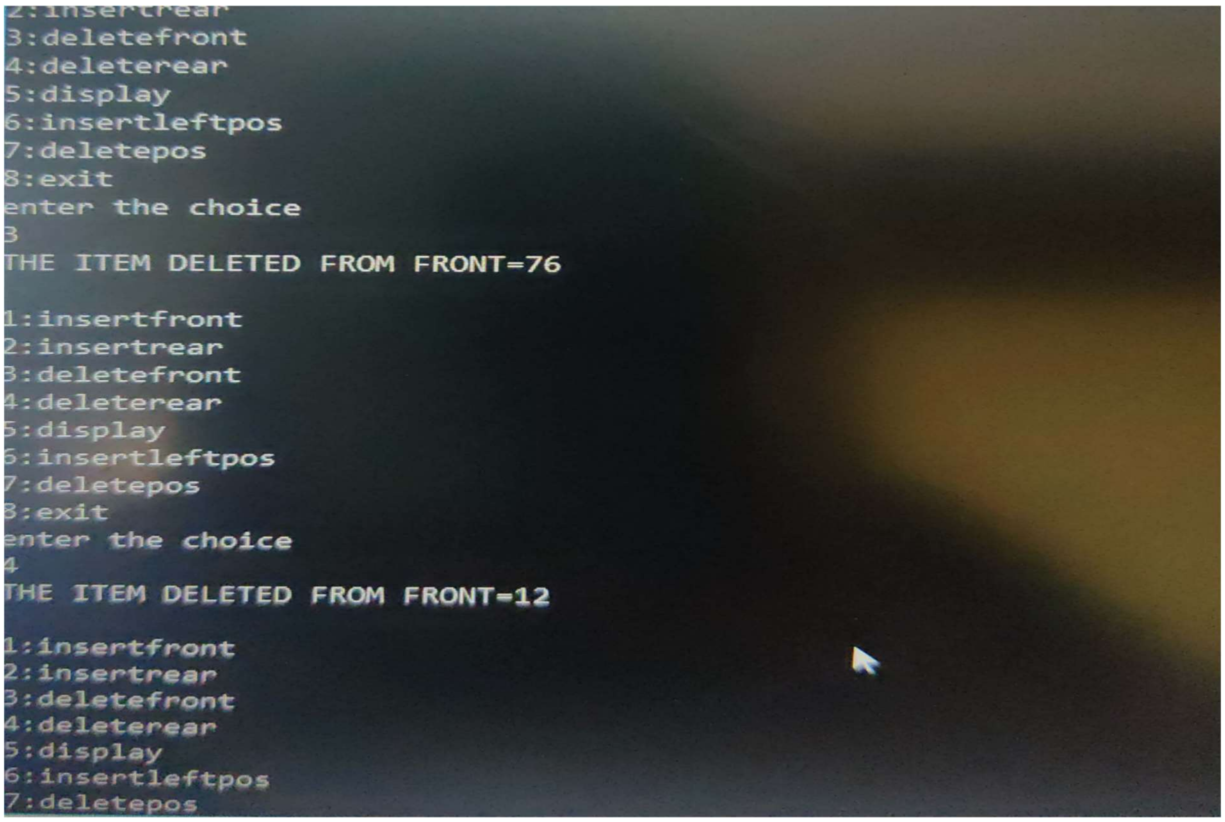
```
1:insertfront
2:insertrear
3:deletefront
4:deleterear
5:display
6:insertleftpos
7:deletepos
8:exit
enter the choice
1
Enter the item at front end:
67

1:insertfront
2:insertrear
3:deletefront
4:deleterear
5:display
6:insertleftpos
7:deletepos
8:exit
enter the choice
6
Enter the key element:
67
Enter the item to be inserted at the left of the given item:
76

1:insertfront
2:insertrear
```



```
3:deletefront
4:deleterear
5:display
6:insertleftpos
7:deletepos
8:exit
enter the choice
5
The contents of the list are:
76
67
12
34

1:insertfront
2:insertrear
3:deletefront
4:deleterear
5:display
6:insertleftpos
7:deletepos
8:exit
enter the choice
1
Enter the item at front end:
34

1:insertfront
2:insertrear
3:deletefront
```



```
4:deleterear
5:display
6:insertleftpos
7:deletepos
8:exit
enter the choice
7
Enter the key element whose duplicates should be removed:
34
Removed all the duplicate elements of the given item successfully

1:insertfront
2:insertrear
3:deletefront
4:deleterear
5:display
6:insertleftpos
7:deletepos
8:exit
enter the choice
5
The contents of the list are:
76
67
12

1:insertfront
2:insertrear
3:deletefront
4:deleterear
```



```
2:insertrear
3:deletefront
4:deleterear
5:display
6:insertleftpos
7:deletepos
8:exit
enter the choice
3
THE ITEM DELETED FROM FRONT=76

1:insertfront
2:insertrear
3:deletefront
4:deleterear
5:display
6:insertleftpos
7:deletepos
8:exit
enter the choice
4
THE ITEM DELETED FROM FRONT=12

1:insertfront
2:insertrear
3:deletefront
4:deleterear
5:display
6:insertleftpos
7:deletepos
```

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

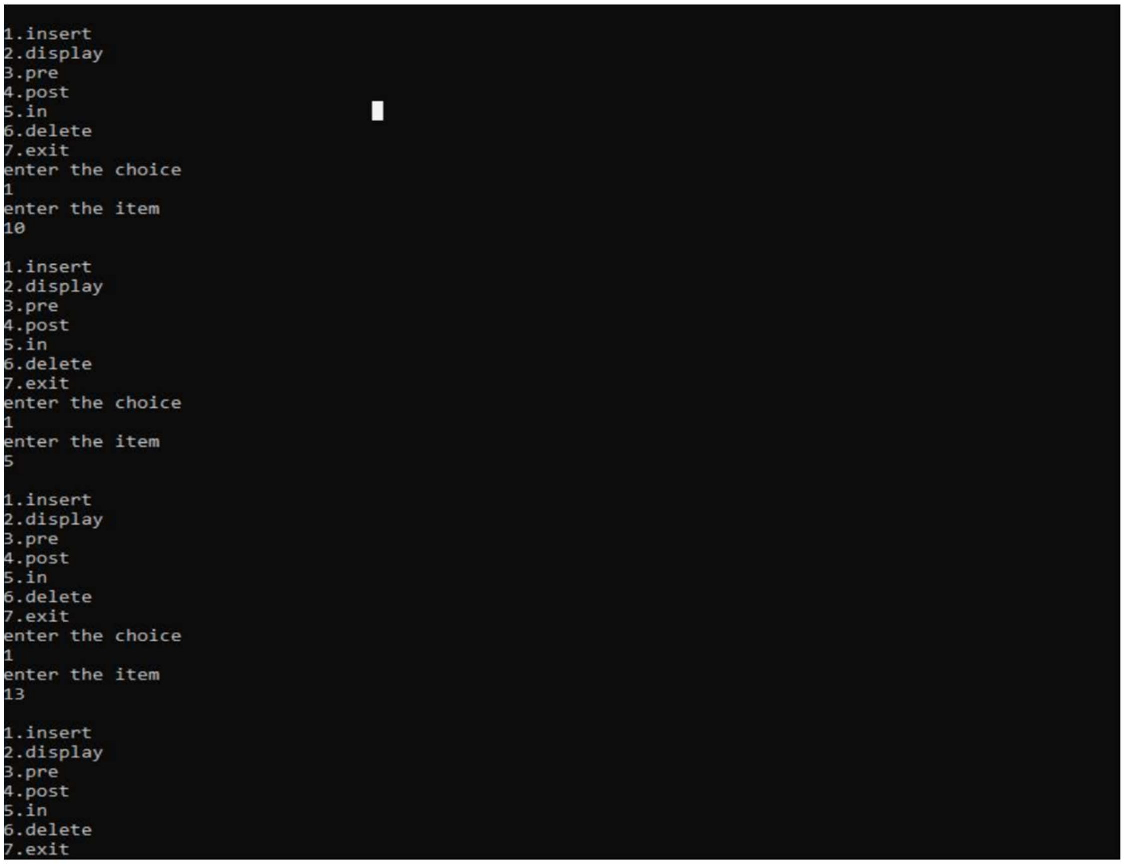
c) To display the elements in the tree.

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

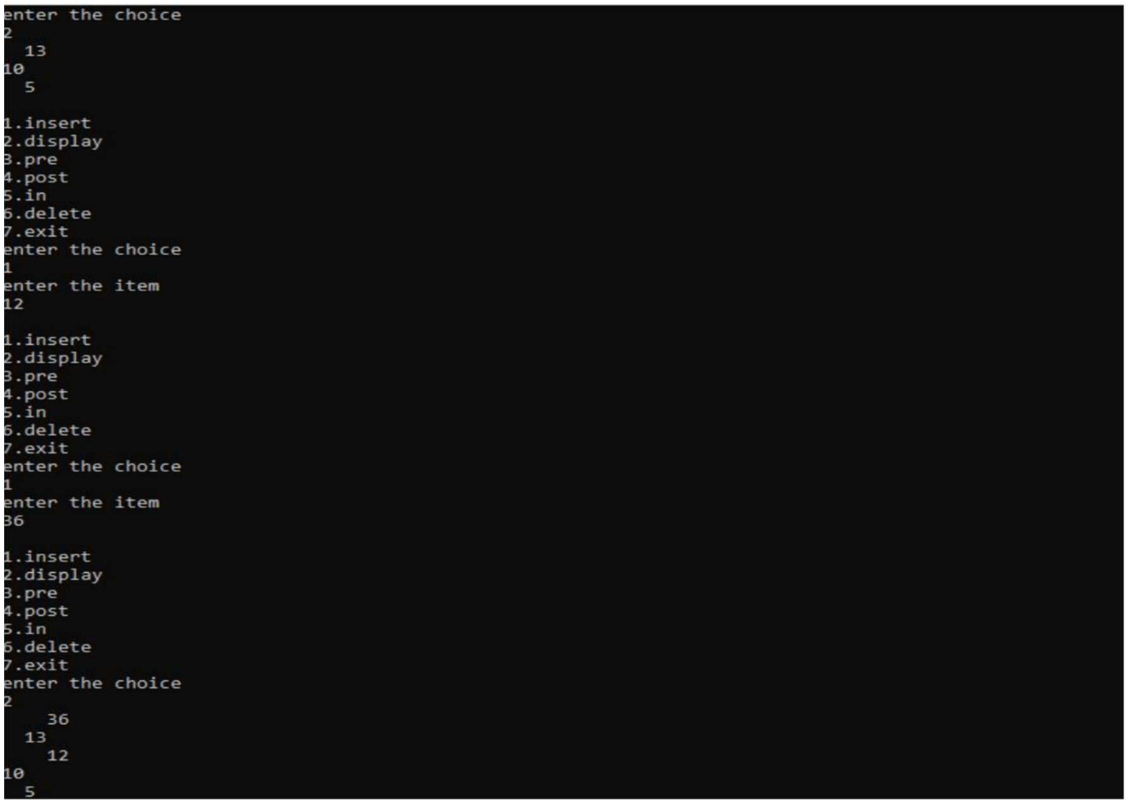
```
cur=root;
while(cur!=NULL)
{
prev=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
 prev->llink=temp;
else
 prev->rlink=temp;
return root;
}
void display(NODE root,int i)
{
int j;
if(root!=NULL)
 {
  display(root->rlink,i+1);
  for(j=0;j<i;j++)
      printf("  ");
   printf("%d\n",root->info);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

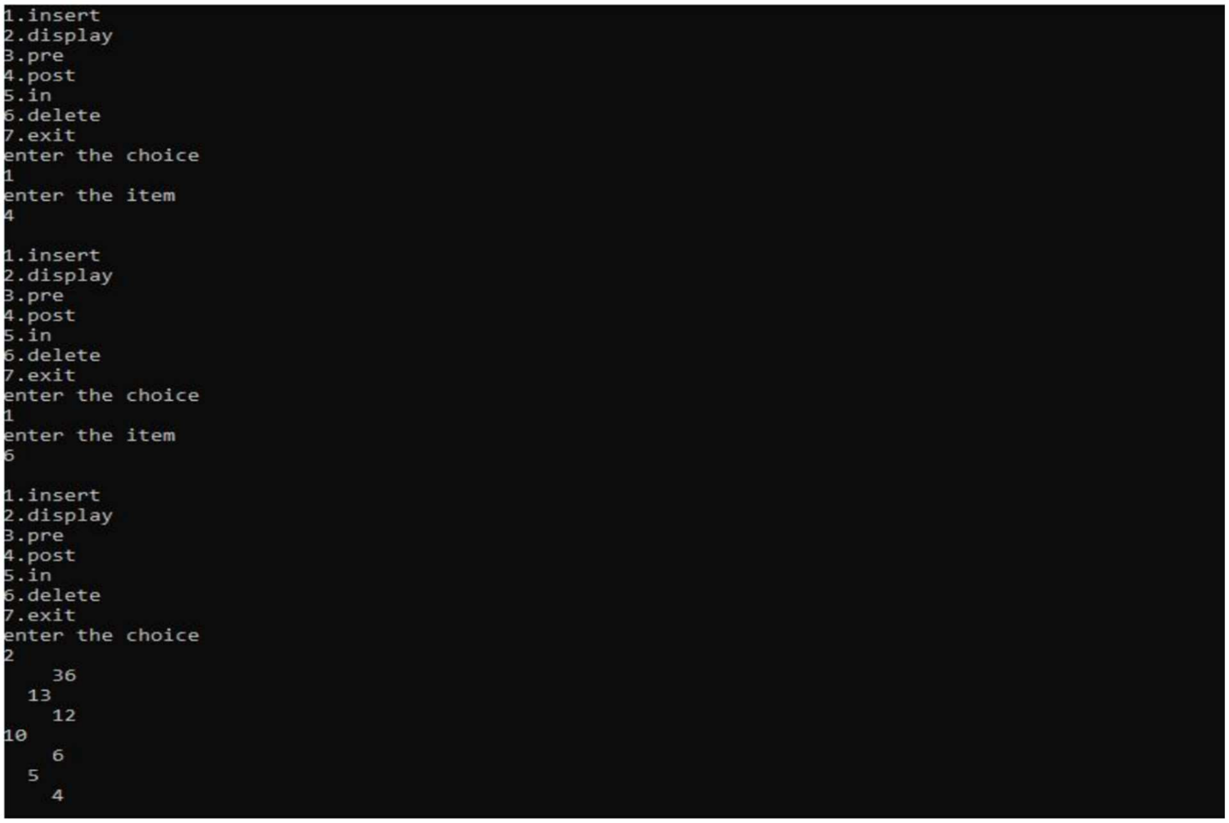
Output:



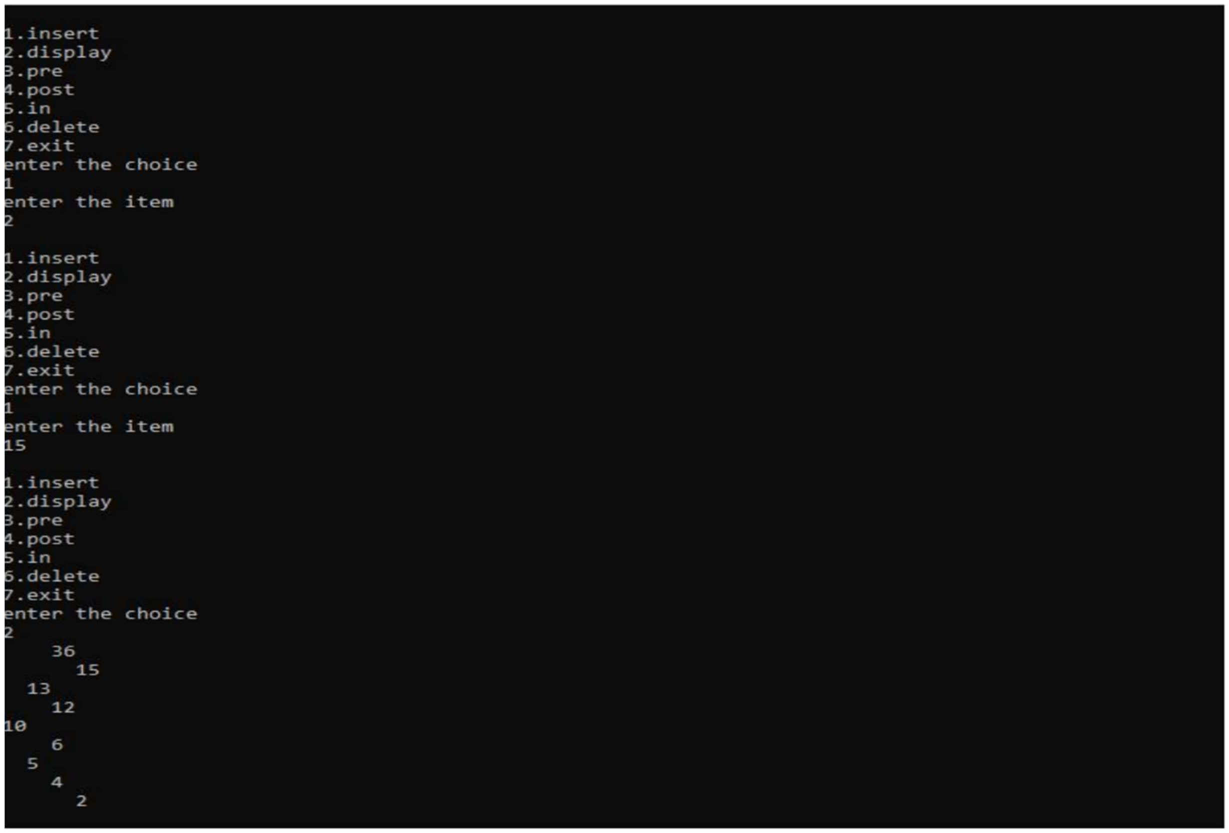
```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
10

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
```



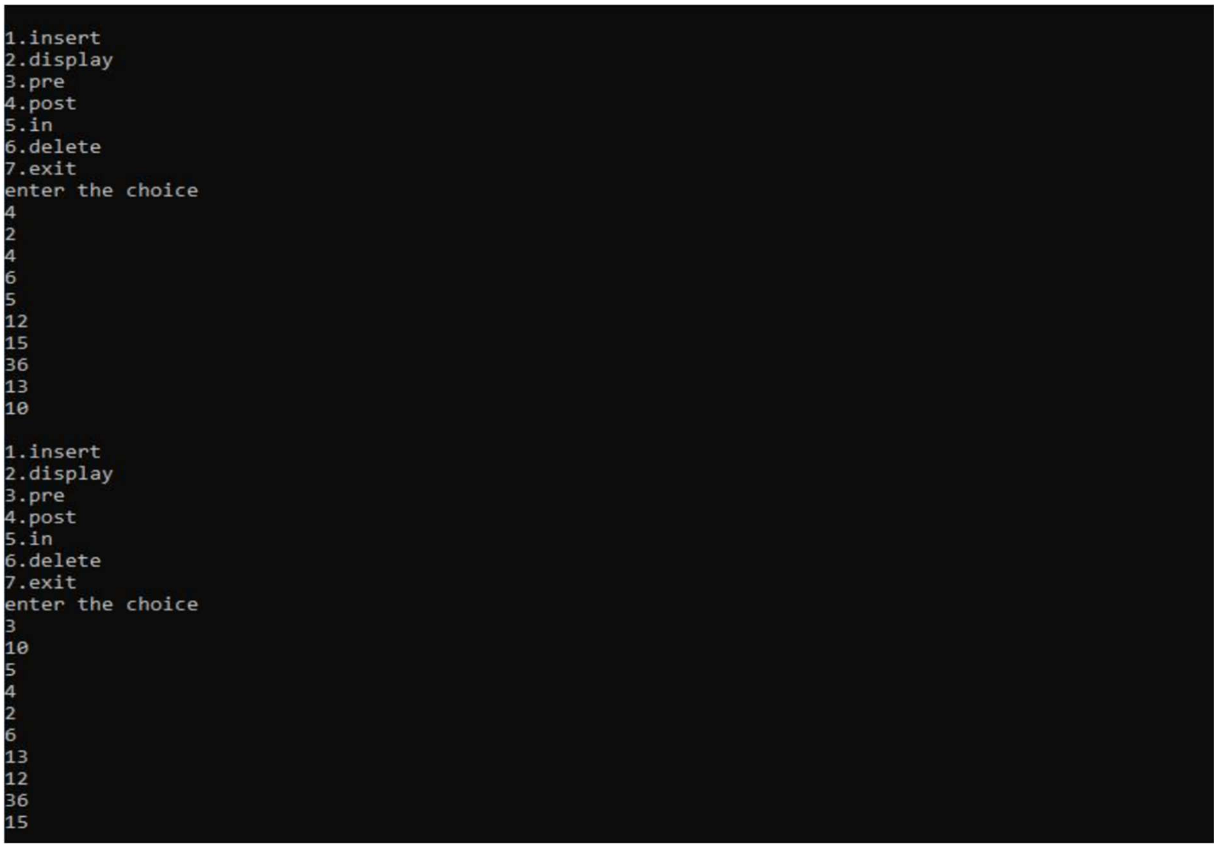
```
enter the choice
2
  13
10
  5

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
```



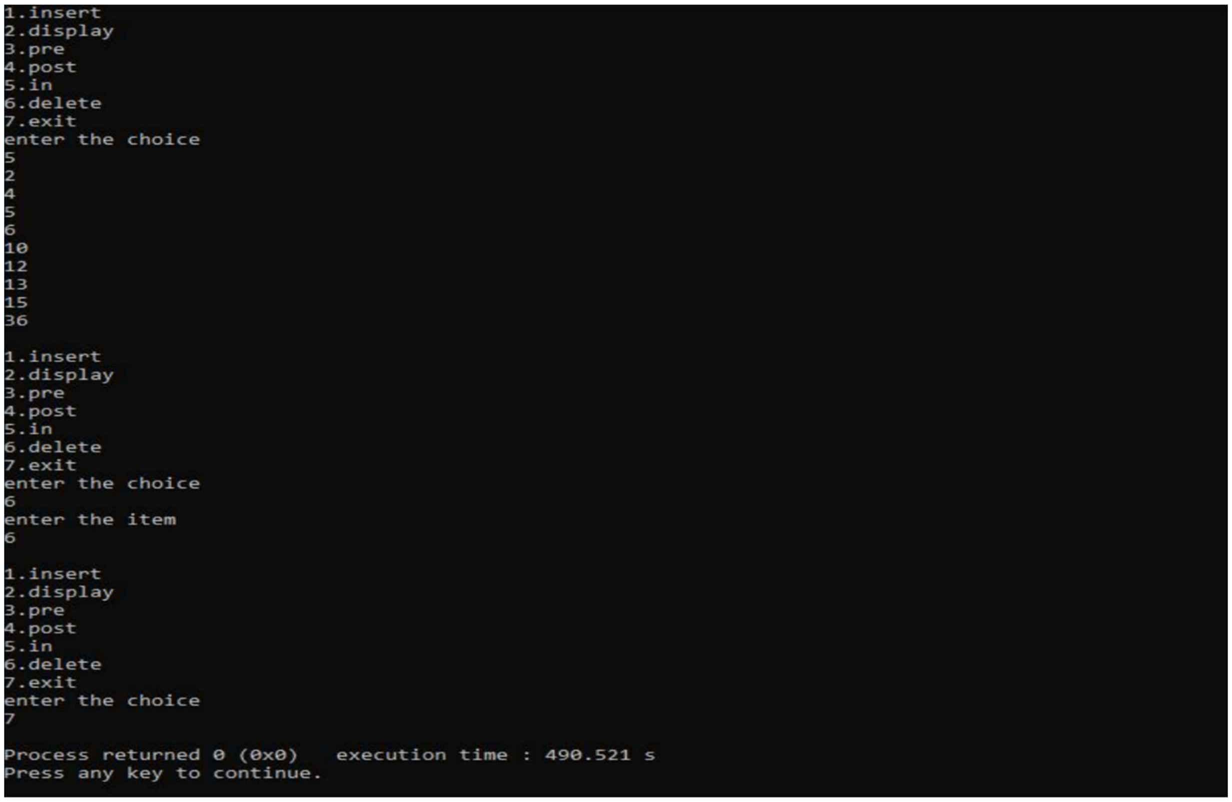
```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
4

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

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



```
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.
```