VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

DATA STRUCTURE LAB RECORD

Submitted by

S Skanda(1BM19CS137)

Under the Guidance of

Prof. Lohith JJ
Assistant Professor, BMSCE

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Sep-2020 to Jan-2021

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Data structures lab carried out by **S Skanda (1BM19CS137)** who is the bonafide students of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visveswaraiah Technological University, Belgaum during the year 2020-2021. The lab report has been approved as it satisfies the academic requirements in respect of **DATA STRUCTURE LAB RECORD (19CS3PCDST)** work prescribed for the said degree.

Signature of the Guide
Prof. Lohith JJ
Professor
BMSCE, Bengaluru

Signature of the HOD
Dr. Umadevi V
Associate Prof.& Head, Dept. of CSE
BMSCE, Bengaluru

Name of the Examiner

Signature with date

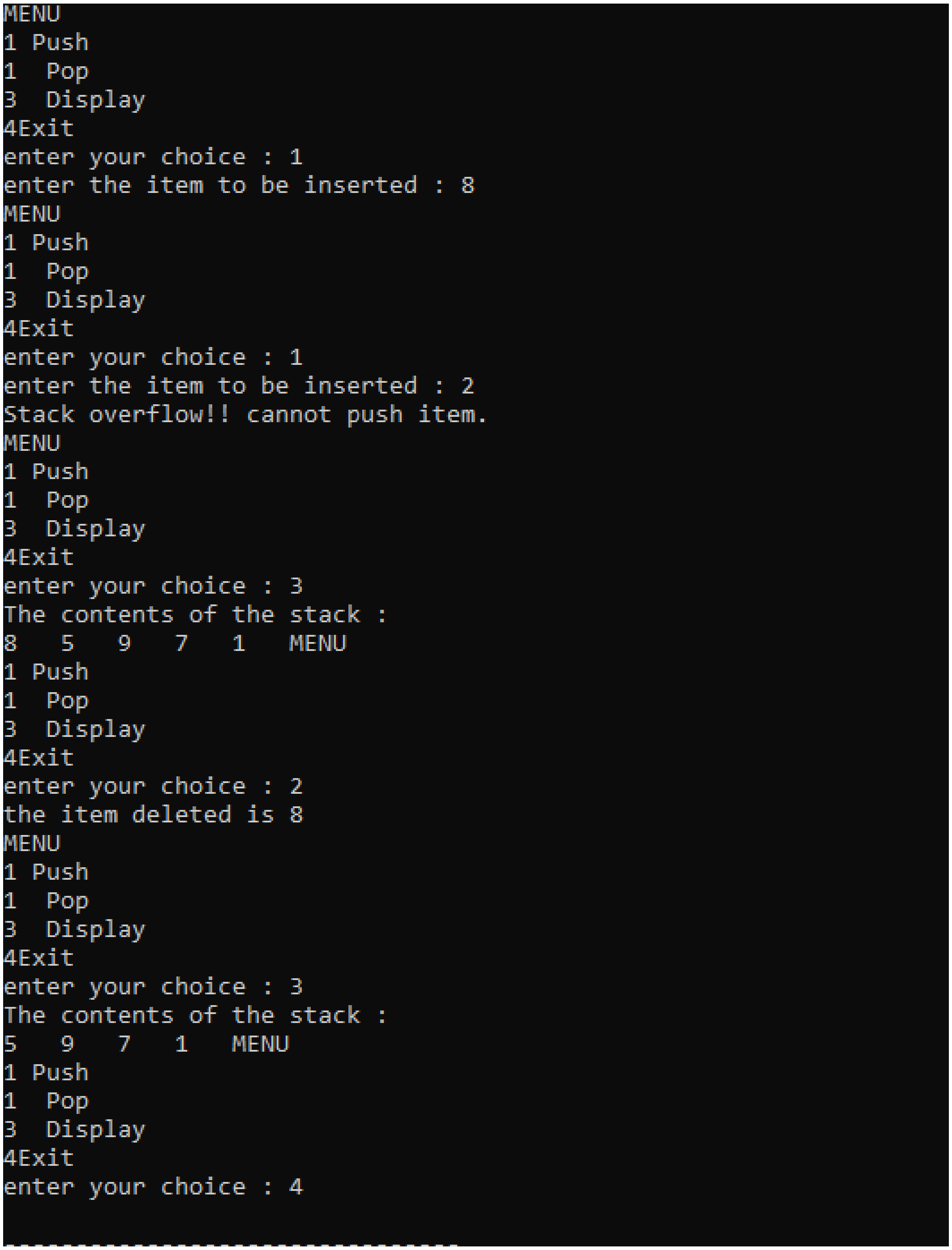
1.______________________________ ________________________

2. ______________________________ ________________________

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>
#define stack_size 5
int top =-1;
int s[10],item;
void push()
{
     if (top==stack_size-1)
     {
          printf("Stack overflow!! cannot push item. \n");
          return ;
     }
     top=top+1;
     s[top]=item;
}
int pop()
{
     if (top== -1)
     {
          return -1;
     }

     return s[top--];
}
void display()
{
     int i;
     if(top==-1)
```

```
    {
        printf("Stack is empty. \n");
    }
    printf("The contents of the stack : \n");
    for(i=tpo;i>=0;i--)
        printf("%d   ",s[i]);
}
int main()
{
    int deleted, choice;
    for(;;)
    {
        printf("MENU \n1 Push\n1  Pop \n3  Display \n4Exit \n");
        printf("enter your choice : ");
        scanf("%d",&choice);
        switch (choice)
        {
            case 1:
                printf("enter the item to be inserted : ");
                scanf("%d",&item);
                push();
                break;
            case 2:
                deleted=pop();
                if(deleted==-1)
                    printf("Stack underflow!! cannot pop item. \n");
                else
                    printf("the item deleted is %d \n",deleted);
                break;
            case 3 :
                display();
                break;
            default :
                exit(0);
        }
    }
}
```

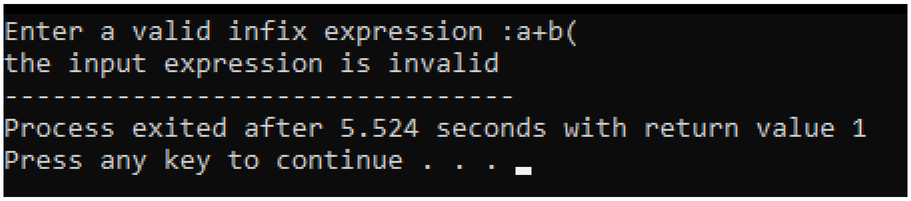


```
MENU
1 Push
1  Pop
3  Display
4Exit
enter your choice : 1
enter the item to be inserted : 8
MENU
1 Push
1  Pop
3  Display
4Exit
enter your choice : 1
enter the item to be inserted : 2
Stack overflow!! cannot push item.
MENU
1 Push
1  Pop
3  Display
4Exit
enter your choice : 3
The contents of the stack :
8   5   9   7   1   MENU
1 Push
1  Pop
3  Display
4Exit
enter your choice : 2
the item deleted is 8
MENU
1 Push
1  Pop
3  Display
4Exit
enter your choice : 3
The contents of the stack :
5   9   7   1   MENU
1 Push
1  Pop
3  Display
4Exit
enter your choice : 4

--------------------------------
```

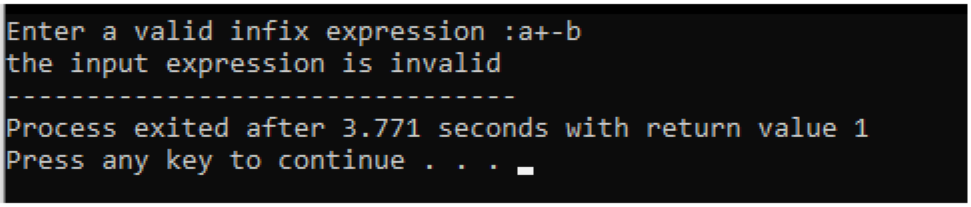
2 WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide)

```
#include<stdio.h>
#include<string.h>
#include<process.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 i,j,top;
```

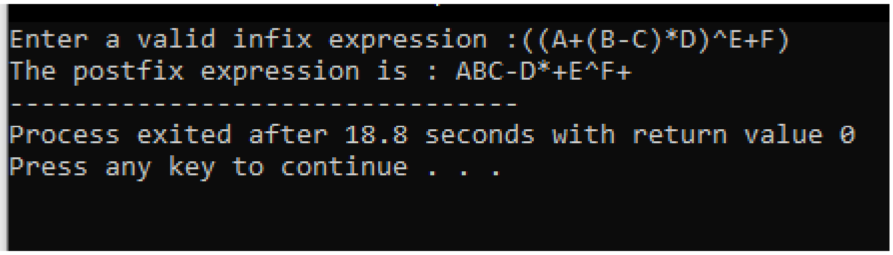
```
    char s[30],symbol;
    top=-1;
    s[++top]='#';
    j=0;
    for(i=0;i<strlen(infix);i++)
    {
        symbol=infix[i];
        while(F(s[top])>G(symbol))
        {
            postfix[j]=s[top--];
            j++;
        }
        if(F(s[top]) != G(symbol))
            s[++top]=symbol;
        else
            top--;
    }
    while(s[top]!='#')
    {
        postfix[j++]=s[top--];

    }
    postfix[j]='\0';

}
int main()
{
    int i,o=0,c=0,flag1,flag2,j;
    char infix[20],postfix[20];
    char ops[10]="+-*/^&";
    printf("Enter a valid infix expression :");
    scanf("%s",&infix);
    for(i=0;i<strlen(infix);i++)
    {
        if(infix[i]=='(')
            o++;
        if(infix[i]==')')
            c++;
        for(j=0;j<strlen(ops);j++)
        {
            if( infix[i]== ops[j] )
                flag1=1;
            if(infix[i+1]== ops[j])
                flag2=1;
        }
        if(flag1==1&&flag2==1)
            {
                printf("the input expression is invalid");
                exit(1);
            }
    }
    if(o!=c)
        {
            printf("the input expression is invalid");
            exit(1);
        }
    infix_postfix(infix,postfix);
    printf("The postfix expression is : %s",postfix);
    return 0;
}
```



```
Enter a valid infix expression :a+b(
the input expression is invalid
--------------------------------
Process exited after 5.524 seconds with return value 1
Press any key to continue . . .
```



```
Enter a valid infix expression :a+-b
the input expression is invalid
--------------------------------
Process exited after 3.771 seconds with return value 1
Press any key to continue . . .
```



```
Enter a valid infix expression :((A+(B-C)*D)^E+F)
The postfix expression is : ABC-D*+E^F+
--------------------------------
Process exited after 18.8 seconds with return value 0
Press any key to continue . . .
```

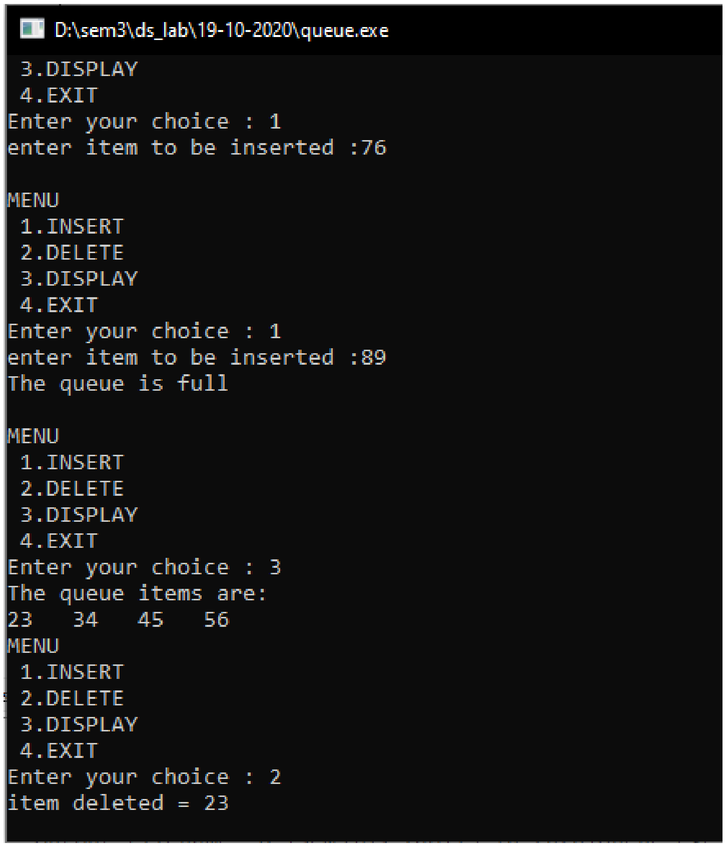
3 WAP to simulate the working of a queue of integers using an array. Provide the following operations

a) Insert b) Delete c) Display

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

```
#include<stdio.h>
#include<process.h>
# define queue_size 5
int item ,front=0, rear=-1, q[10];
void insert()
{
      if(rear == queue_size-1)
      {
            printf("The queue is full \n");
            return ;
      }
      rear+=1;
      q[rear]=item;
}
int delete()
{
      if(front>rear)
      {
            front=0;
            rear=-1;
            return -1;
      }
      return q[front++];
}
void display()
{
      int i;
      if(front>rear)
      {
            printf("The queue is empty");
            return;
      }
      printf("The queue items are: \n");
      for(i=front;i<=rear;i++)
      {
            printf("%d   ",q[i]);
      }
}
int main()
{
      int choice;
      for(;;)
```

```
	{
		printf("\nMENU \n 1.INSERT \n 2.DELETE \n 3.DISPLAY \n 4.EXIT \n");
		printf("Enter your choice : ");
		scanf("%d",&choice);
		switch (choice)
		{
			case 1 :
				printf("enter item to be inserted :");
				scanf("%d",&item);
				insert();
				break;
			case 2 :
				item =delete();
				if(item==-1)
					printf("the queue is empty \n");
				else
					printf("item deleted = %d \n",item);
				break;
			case 3:
				display();
				break;
			default: exit(0);
		}
	}
	return 0;
}
```



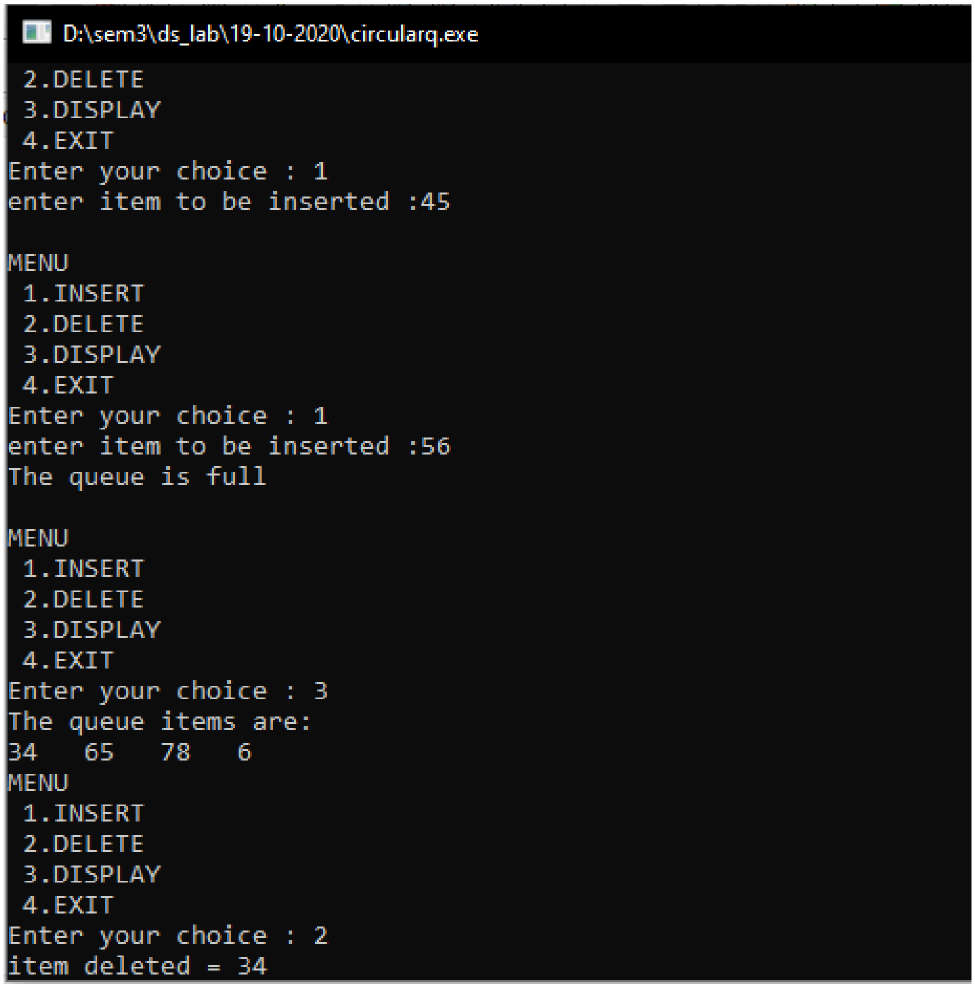
```
 3.DISPLAY
 4.EXIT
Enter your choice : 1
enter item to be inserted :76

MENU
 1.INSERT
 2.DELETE
 3.DISPLAY
 4.EXIT
Enter your choice : 1
enter item to be inserted :89
The queue is full

MENU
 1.INSERT
 2.DELETE
 3.DISPLAY
 4.EXIT
Enter your choice : 3
The queue items are:
23   34   45   56
MENU
 1.INSERT
 2.DELETE
 3.DISPLAY
 4.EXIT
Enter your choice : 2
item deleted = 23
```

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 queue_size 5
int item ,front=0, rear=-1, q[10],count=0;
void insert()
{
      if(count == queue_size)
      {
            printf("The queue is full \n");
            return ;
      }
      rear=(rear+1)%queue_size;
      q[rear]=item;
      count+=1;
}
int delete()
{
      if(count==0)
      {
            front=0;
            rear=-1;
            return -1;
      }
      item=q[front];
      front=(front+1)%queue_size;
      count-=1;
      return item;
}
void display()
{
      int i, f=front;
      if(count==0)
      {
            printf("The queue is empty");
            return;
      }
      printf("The queue items are: \n");
      for(i=1;i<=count;i++)
      {
            printf("%d   ",q[f]);
            f=(f+1)%queue_size;
      }
}
int main()
{
      int choice;
      for(;;)
      {
            printf("\nMENU \n 1.INSERT \n 2.DELETE \n 3.DISPLAY \n 4.EXIT \n");
            printf("Enter your choice : ");
            scanf("%d",&choice);
            switch (choice)
            {
                  case 1 :
                        printf("enter item to be inserted :");
                        scanf("%d",&item);
                        insert();
                        break;
                  case 2 :
                        item =delete();
                        if(item==-1)
                              printf("the queue is empty \n");
                        else
                              printf("item deleted = %d \n",item);
                        break;
                  case 3:
                        display();
                        break;
                  default: exit(0);
            }
      }
      return 0;
}
```



D:\sem3\ds_lab\19-10-2020\circularq.exe

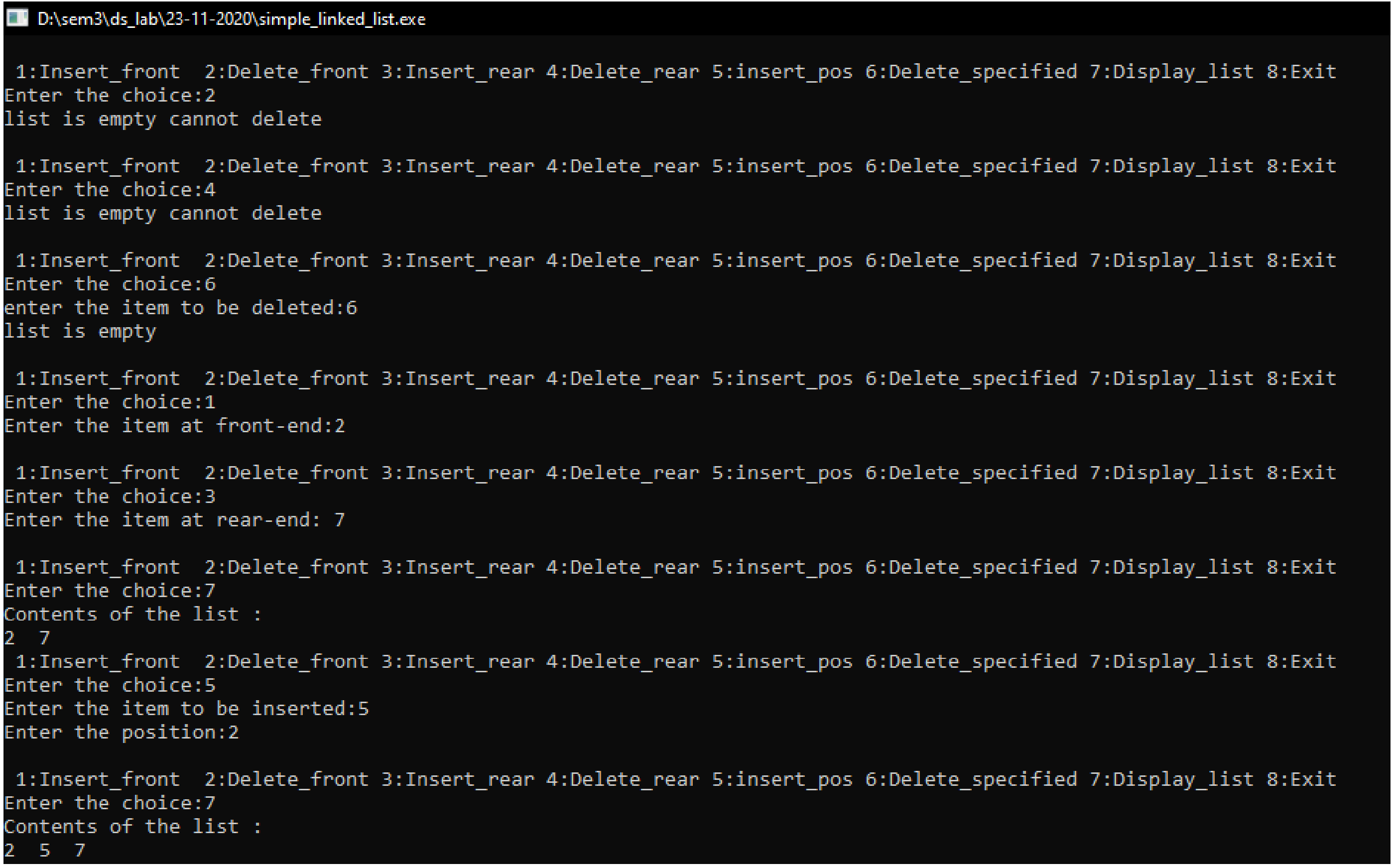
```
 2.DELETE
 3.DISPLAY
 4.EXIT
Enter your choice : 1
enter item to be inserted :45

MENU
 1.INSERT
 2.DELETE
 3.DISPLAY
 4.EXIT
Enter your choice : 1
enter item to be inserted :56
The queue is full

MENU
 1.INSERT
 2.DELETE
 3.DISPLAY
 4.EXIT
Enter your choice : 3
The queue items are:
34   65   78   6
MENU
 1.INSERT
 2.DELETE
 3.DISPLAY
 4.EXIT
Enter your choice : 2
item deleted = 34
```

5 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.
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) Display the contents of the linked list.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<process.h>
struct node
{
int info;
struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("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 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 delete_info(int key,NODE first)
```

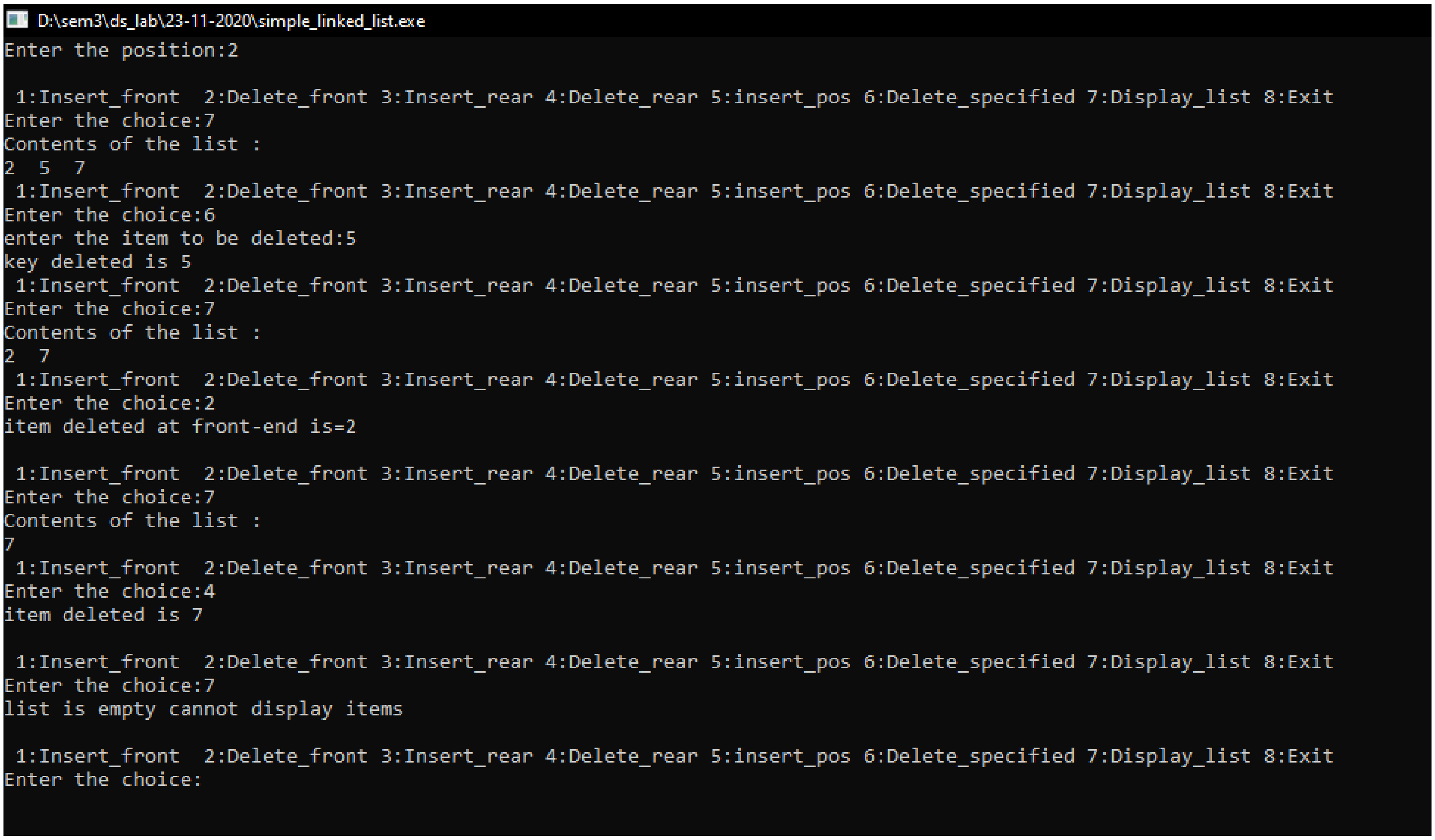
```
{
NODE prev,cur;
if(first==NULL)
{
printf("list is empty\n");
return NULL;
}
if(key==first->info)
{
cur=first;
first=first->link;
freenode(cur);
return first;
}
prev=NULL;
cur=first;
while(cur!=NULL)
{
if(key==cur->info)break;
prev=cur;
cur=cur->link;
}
if(cur==NULL)
{
printf("search is unsuccessfull\n");
return first;
}
prev->link=cur->link;
printf("key deleted is %d",cur->info);
freenode(cur);
return first;
}
NODE insert_pos( int item, int pos, NODE first)
{
NODE temp;
NODE prev,cur;
int count;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL && pos==1)
{
return temp;
}
if(first==NULL)
{
printf("invalid position\n");
return first;
}
if(pos==1)
{
```

```
temp->link=first;
return temp;
}
count=1;
prev=NULL;
cur=first;
while(cur!=NULL && count!=pos)
{
prev=cur;
cur=cur->link;
count++;
}
if(count==pos)
{
prev->link=temp;
temp->link=cur;
return first;
}
printf("invalid position\n");
return first;
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list is empty cannot display items\n");
else
{
printf("Contents of the list : \n");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d ",temp->info);
}
}
}
int main()
{
int item,choice,key,pos;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_front 2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos
6:Delete_specified 7:Display_list 8:Exit\n");
printf("Enter the choice:");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("Enter the item at front-end:");
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: ");
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:");
scanf("%d",&item);
printf("Enter the position:");
scanf("%d",&pos);
insert_pos( item, pos, first);
break;
case 6:printf("enter the item to be deleted:");
scanf("%d",&key);
first=delete_info(key,first);
break;
case 7:display(first);
break;
default:exit(0);
break;
}
}
getch();
return 0;
} ..
```



D:\sem3\ds_lab\23-11-2020\simple_linked_list.exe

```
 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:2
list is empty cannot delete

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:4
list is empty cannot delete

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:6
enter the item to be deleted:6
list is empty

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8: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:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:3
Enter the item at rear-end: 7

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:7
Contents of the list :
2  7
 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:5
Enter the item to be inserted:5
Enter the position:2

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:7
Contents of the list :
2  5  7
```



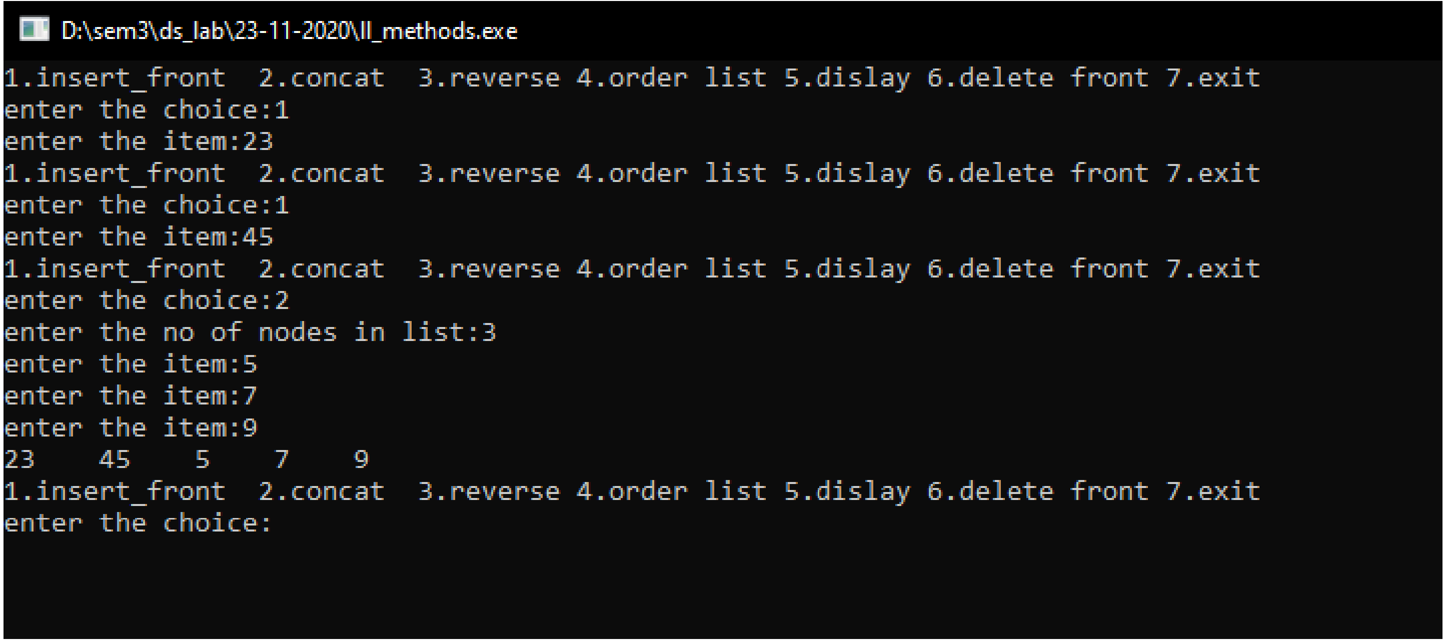
```
Enter the position:2

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:7
Contents of the list :
2  5  7
 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:6
enter the item to be deleted:5
key deleted is 5
 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:7
Contents of the list :
2  7
 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:2
item deleted at front-end is=2

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:7
Contents of the list :
7
 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:4
item deleted is 7

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:7
list is empty cannot display items

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:
```

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

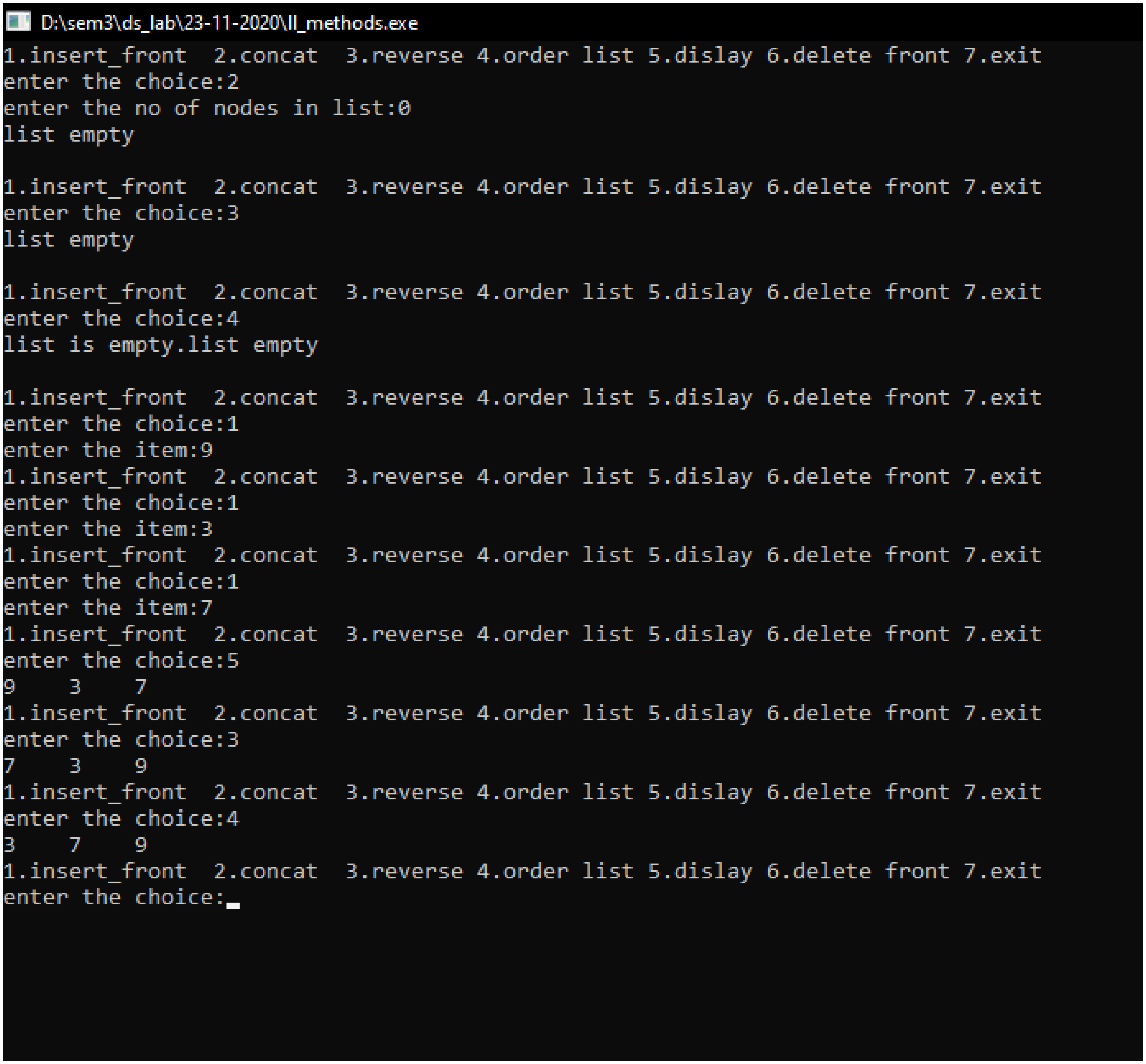
```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<process.h>
struct node
 {
  int info;
  struct node *link;
 };
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("mem full\n");
  exit(0);
 }
 return x;
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
 return temp;
cur=first;
while(cur->link!=NULL)
 cur=cur->link;
cur->link=temp;
return first;
}
NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",first->info);
```

```
free(first);
return temp;
}

void display(NODE first)
{
 NODE temp;
 if(first==NULL)
  printf("list empty \n");

 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d    ",temp->info);
  }
  printf("\n");
}
NODE concat(NODE first,NODE second)
{
 NODE cur;
 if(first==NULL)
  return second;
 if(second==NULL)
  return first;
 cur=first;
 while(cur->link!=NULL)
  cur=cur->link;
 cur->link=second;
 return first;
}
NODE reverse(NODE first)
 {
 NODE cur,temp;
 cur=NULL;
 while(first!=NULL)
  {
   temp=first;
   first=first->link;
   temp->link=cur;
   cur=temp;
  }
 return cur;
}
    NODE sortList(NODE first) {
        NODE current = first, index = NULL;
        int temp;

        if(first == NULL) {
           printf("list is empty.");
            return current;
        }
        else {
            while(current != NULL) {

                index = current->link;

                while(index != NULL) {

                    if(current->info > index->info) {
                        temp = current->info;
                        current->info = index->info;
                        index->info = temp;
                    }
                    index = index->link;
                }
                current = current->link;
            }
                return current;
        }
    }

int main()
{
int item,choice,pos,i,n;
NODE first=NULL,a,b;
for(;;)
{
printf("1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 
7.exit\n");
printf("enter the choice:");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item:");
         scanf("%d",&item);
         first=insert_rear(first,item);
         break;
  case 2:printf("enter the no of nodes in list:");
         scanf("%d",&n);
         a=NULL;
         for(i=0;i<n;i++)
          {
           printf("enter the item:");
           scanf("%d",&item);
           a=insert_rear(a,item);
          }
          first=concat(first,a);
          display(first);
         break;
case 3:first=reverse(first);
       display(first);
       break;
case 4:sortList(first);
          display(first);
       break;
case 5:display(first);
       break;
case 6:first=delete_front(first);
   break;
default:exit(0);
}
}
return 0;
}
```



D:\sem3\ds_lab\23-11-2020\ll_methods.exe

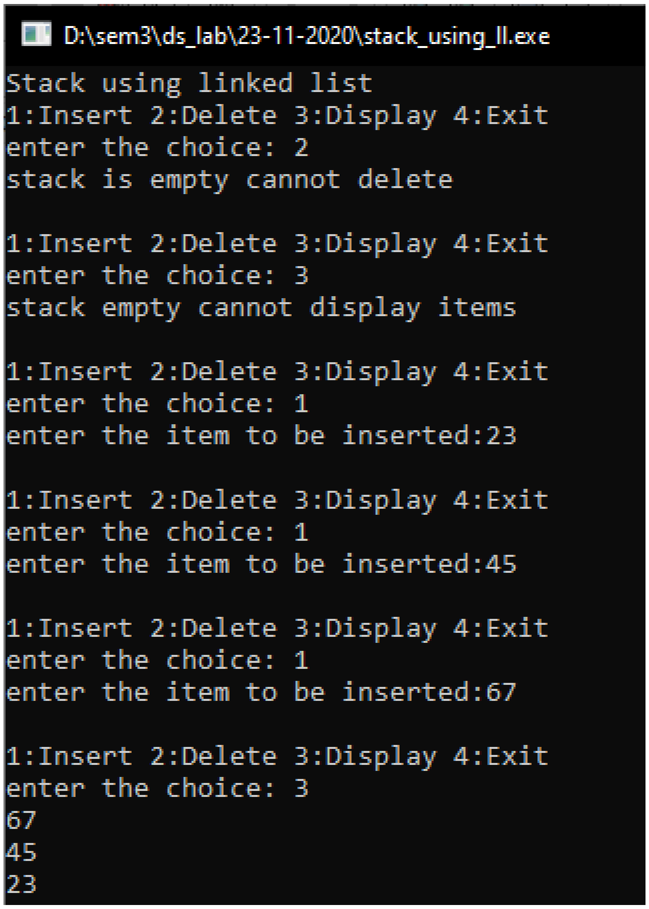
```
1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:1
enter the item:23
1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:1
enter the item:45
1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:2
enter the no of nodes in list:3
enter the item:5
enter the item:7
enter the item:9
23    45    5    7    9
1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:
```



```
1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:2
enter the no of nodes in list:0
list empty

1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:3
list empty

1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:4
list is empty.list empty

1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:1
enter the item:9
1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:1
enter the item:3
1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:1
enter the item:7
1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:5
9    3    7
1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:3
7    3    9
1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:4
3    7    9
1.insert_front  2.concat  3.reverse 4.order list 5.dislay 6.delete front 7.exit
enter the choice:
```

8 WAP to implement Stack & Queues using Linked Representation

```
/*stack using linked list*/
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<process.h>
struct node
{
  int info;
  struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("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 delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("stack is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted is=%d\n",first->info);
free(first);
return temp;
}
void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("stack empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\n",temp->info);
  }
}
void main()
{
int item,choice,pos;
NODE first=NULL;
printf("Stack using linked list");
for(;;)
{
printf("\n1:Insert 2:Delete 3:Display 4:Exit \n");
printf("enter the choice: ");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item to be inserted:");
     scanf("%d",&item);
     first=insert_front(first,item);
     break;
  case 2:first=delete_front(first);
     break;
  case 3:display(first);
     break;
 default:exit(0);
     break;
 }
}
}
```

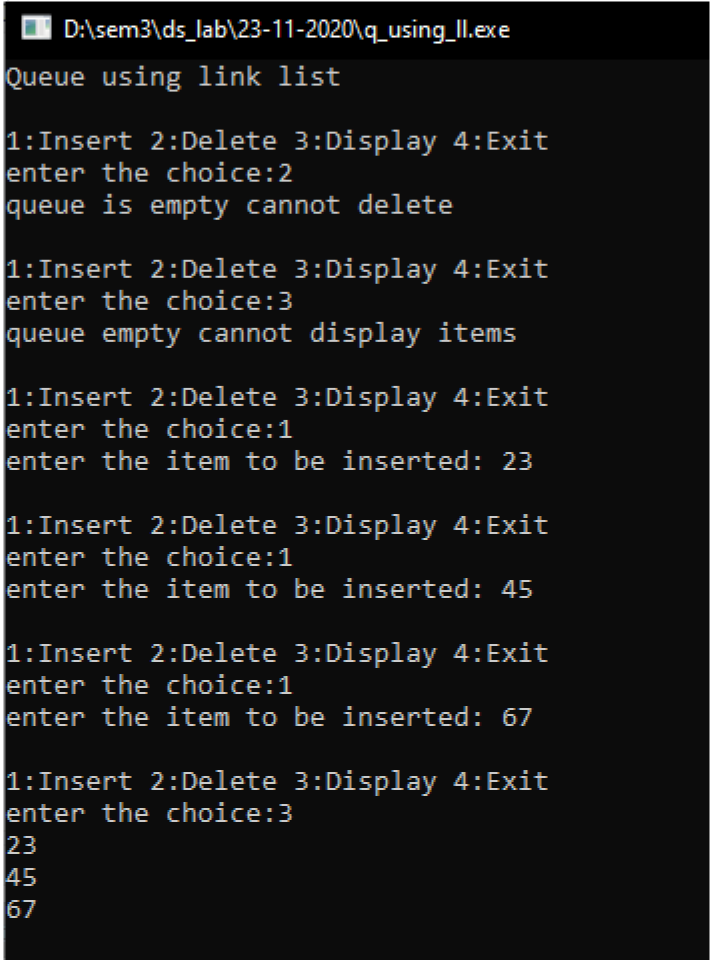


D:\sem3\ds_lab\23-11-2020\stack_using_ll.exe

```
Stack using linked list
1:Insert 2:Delete 3:Display 4:Exit
enter the choice: 2
stack is empty cannot delete

1:Insert 2:Delete 3:Display 4:Exit
enter the choice: 3
stack empty cannot display items

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:45

1:Insert 2:Delete 3:Display 4:Exit
enter the choice: 1
enter the item to be inserted:67

1:Insert 2:Delete 3:Display 4:Exit
enter the choice: 3
67
45
23
```

```
/*quque using linked list*/
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<process.h>
struct node
{
  int info;
  struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("memory full\n");
  exit(0);
 }
 return x;
}
void freenode(NODE x)
{
```

```
free(x);
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
 return temp;
cur=first;
while(cur->link!=NULL)
 cur=cur->link;
cur->link=temp;
return first;
}

NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("queue is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted is=%d\n",first->info);
free(first);
return temp;
}
void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("queue empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\n",temp->info);
  }
}
void main()
{
int item,choice,pos;
NODE first=NULL;
printf("Queue using link list\n");
for(;;)
{
printf("\n1:Insert 2:Delete 3:Display 4:Exit\n");
printf("enter the choice:");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item to be inserted: ");
     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;
 }
}
getch();
}
```



D:\sem3\ds_lab\23-11-2020\q_using_ll.exe

```
Queue using link list

1:Insert 2:Delete 3:Display 4:Exit
enter the choice:2
queue is empty cannot delete

1:Insert 2:Delete 3:Display 4:Exit
enter the choice:3
queue empty cannot display items

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: 45

1:Insert 2:Delete 3:Display 4:Exit
enter the choice:1
enter the item to be inserted: 67

1:Insert 2:Delete 3:Display 4:Exit
enter the choice:3
23
45
67
```

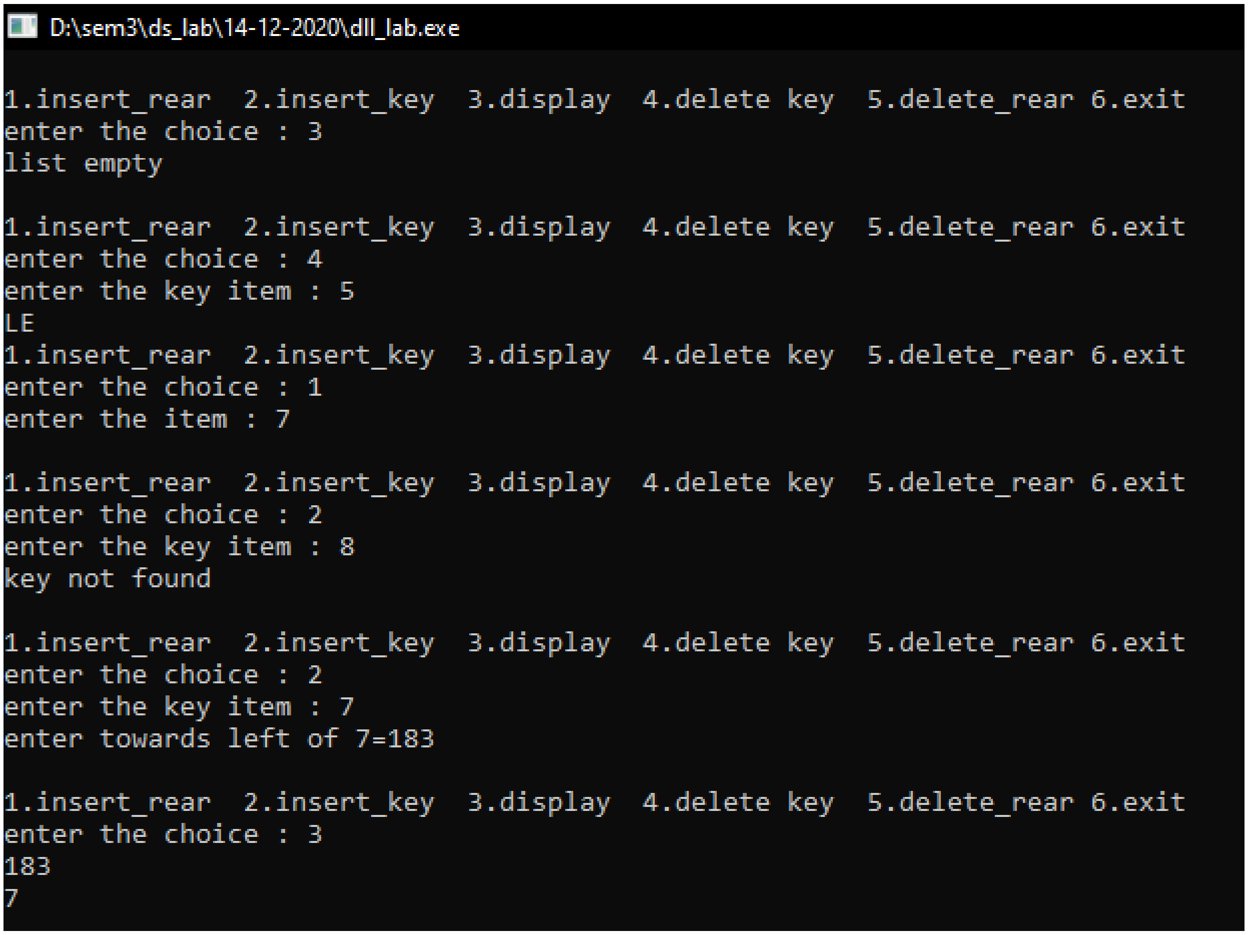
9 WAP Implement doubly link list with primitive operations
a) a) Create a doubly linked list. b) Insert a new node to the left of the node.
b) c) Delete the node based on a specific value. c) Display the contents of the list

```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
struct node
 {
  int info;
  struct node *rlink;
  struct node *llink;
 };
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("mem full\n");
  exit(0);
 }
 return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert_rear(NODE head,int item)
{
NODE temp,cur;
temp=getnode();
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
cur=head->llink;
temp->llink=cur;
cur->rlink=temp;
head->llink=temp;
temp->rlink=head;
head->info=head->info+1;
return head;
}
NODE insert_leftpos(int item,NODE head)
{
NODE temp,cur,prev;
if(head->rlink==head)
{
```

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

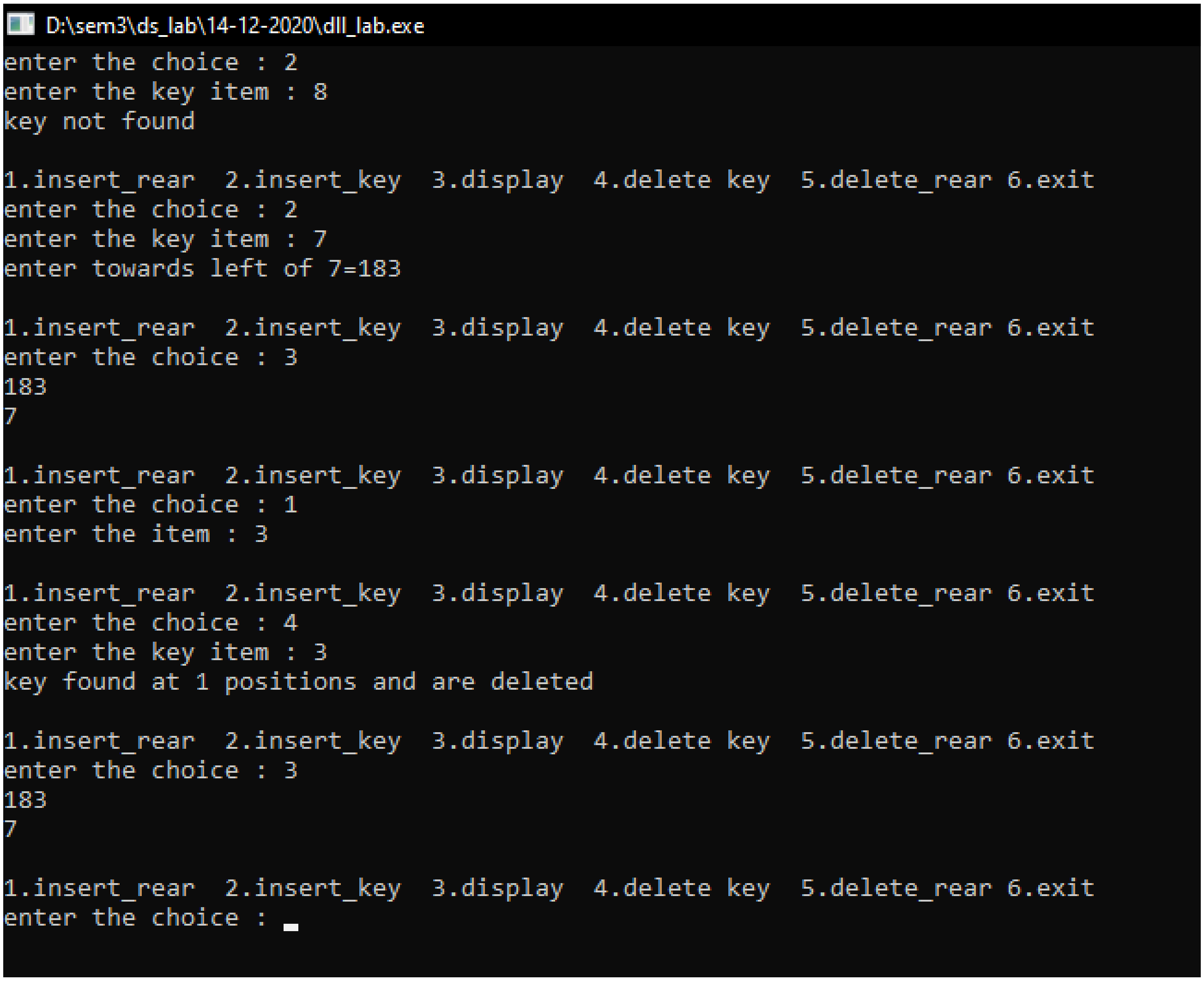
```
}
if(count==0)
  printf("key not found");
  else
 printf("key found at %d positions and are deleted\n", count);

return head;
}
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;
head->llink=prev;
prev->rlink=head;
printf("the node deleted is %d \n",cur->info);
freenode(cur);
return head;
}
void display(NODE head)
{
NODE temp;
if(head->rlink==head)
{
printf("list empty\n");
return;
}
for(temp=head->rlink;temp!=head;temp=temp->rlink)
printf("%d\n",temp->info);
}
void main()
{
int item,choice,key;
NODE head,tem;
head=getnode();
head->rlink=head;
head->llink=head;
for(;;)
{
printf("\n1.insert_rear  2.insert_key  3.display  4.delete key  5.delete_rear
6.exit\n");
printf("enter the choice : ");
scanf("%d",&choice);
switch(choice)
```

```
 {
  case 1:printf("enter the item : ");
            scanf("%d",&item);
            head=insert_rear(head,item);
            break;
  case 2:printf("enter the key item : ");
            scanf("%d",&item);
            head=insert_leftpos(item,head);
            break;
  case 3:display(head);
            break;
  case 4:printf("enter the key item : ");
            scanf("%d",&item);
            head=delete_all_key(item,head);
            break;
  case 5:head=ddelete_rear(head);
                 break;
  default:exit(0);
             break;
  }
 }
}
```



D:\sem3\ds_lab\14-12-2020\dll_lab.exe

```

1.insert_rear  2.insert_key  3.display  4.delete key  5.delete_rear 6.exit
enter the choice : 3
list empty

1.insert_rear  2.insert_key  3.display  4.delete key  5.delete_rear 6.exit
enter the choice : 4
enter the key item : 5
LE
1.insert_rear  2.insert_key  3.display  4.delete key  5.delete_rear 6.exit
enter the choice : 1
enter the item : 7

1.insert_rear  2.insert_key  3.display  4.delete key  5.delete_rear 6.exit
enter the choice : 2
enter the key item : 8
key not found

1.insert_rear  2.insert_key  3.display  4.delete key  5.delete_rear 6.exit
enter the choice : 2
enter the key item : 7
enter towards left of 7=183

1.insert_rear  2.insert_key  3.display  4.delete key  5.delete_rear 6.exit
enter the choice : 3
183
7
```



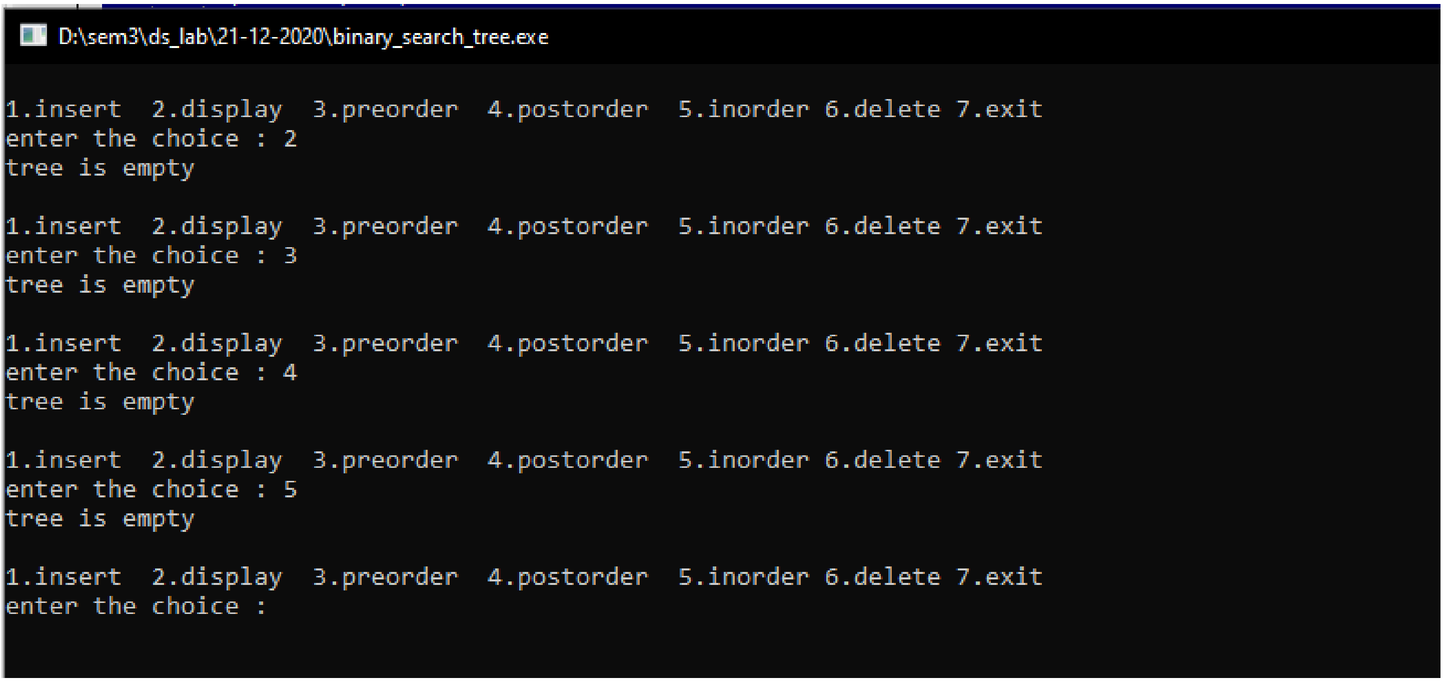
```
enter the choice : 2
enter the key item : 8
key not found

1.insert_rear  2.insert_key  3.display  4.delete key  5.delete_rear 6.exit
enter the choice : 2
enter the key item : 7
enter towards left of 7=183

1.insert_rear  2.insert_key  3.display  4.delete key  5.delete_rear 6.exit
enter the choice : 3
183
7

1.insert_rear  2.insert_key  3.display  4.delete key  5.delete_rear 6.exit
enter the choice : 1
enter the item : 3

1.insert_rear  2.insert_key  3.display  4.delete key  5.delete_rear 6.exit
enter the choice : 4
enter the key item : 3
key found at 1 positions and are deleted

1.insert_rear  2.insert_key  3.display  4.delete key  5.delete_rear 6.exit
enter the choice : 3
183
7

1.insert_rear  2.insert_key  3.display  4.delete key  5.delete_rear 6.exit
enter the choice : _
```

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.

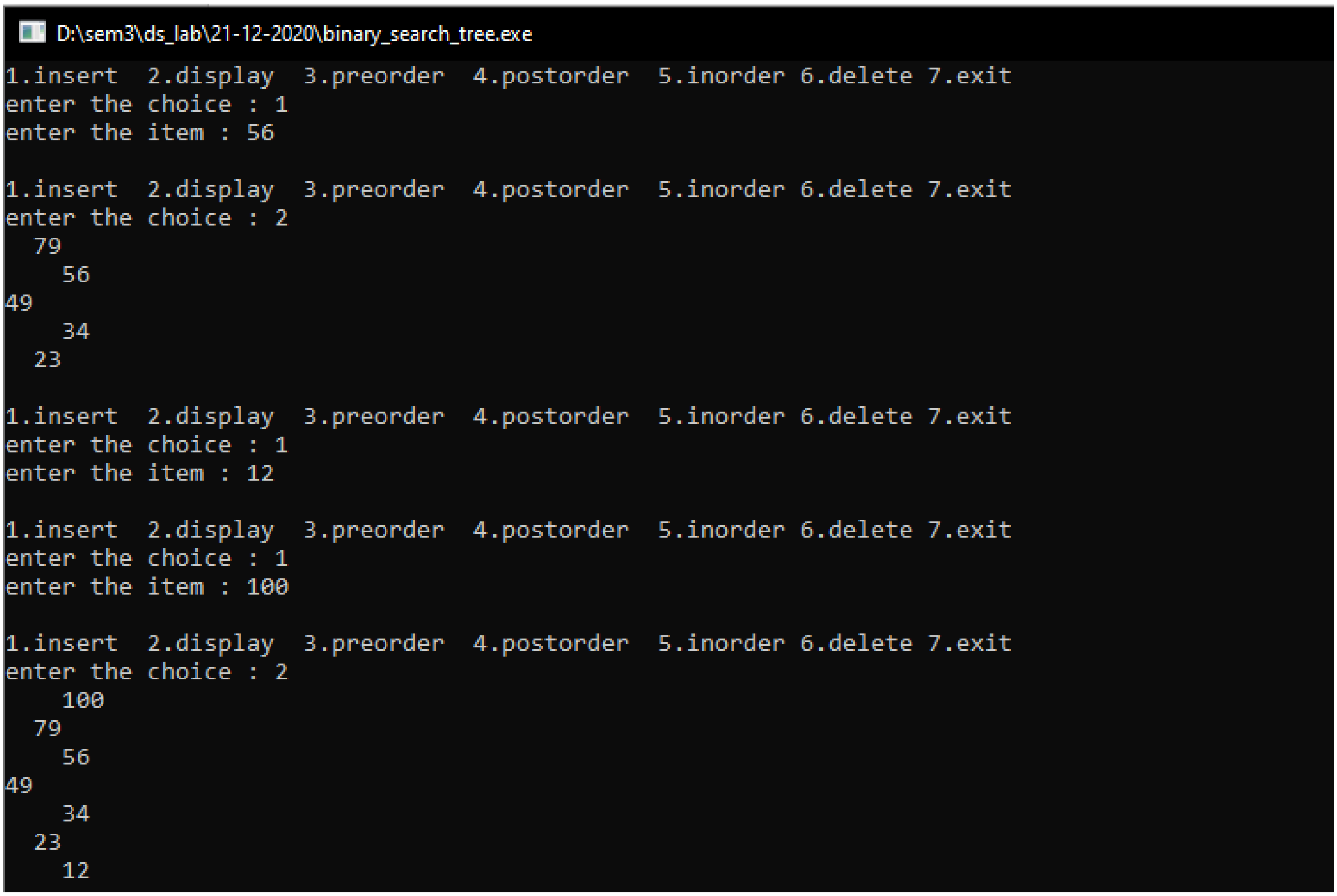
```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
struct node
 {
  int info;
  struct node *rlink;
  struct node *llink;
 };
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("memory full\n");
  exit(0);
 }
 return x;
}
void freenode(NODE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

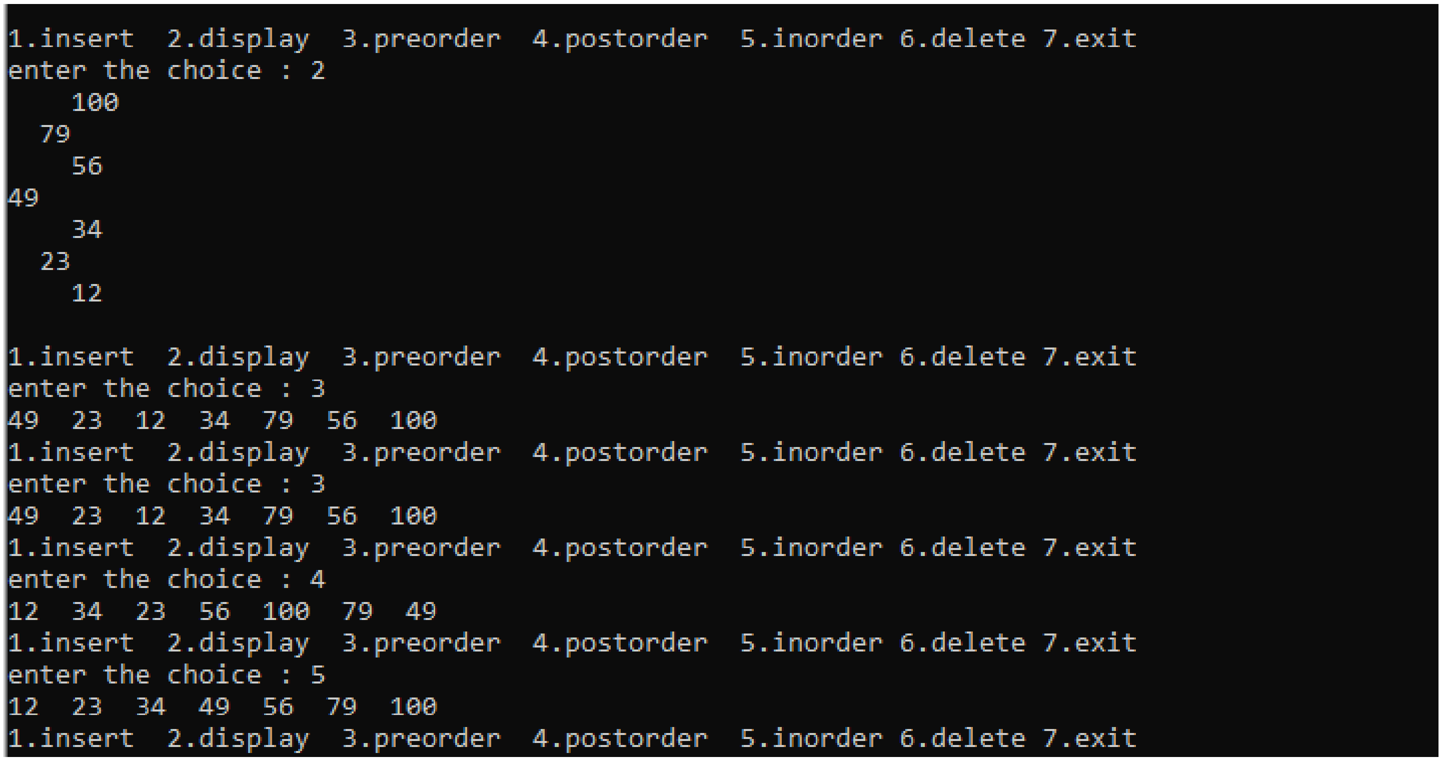
```
  suc=suc->llink;
 suc->llink=cur->llink;
 q=cur->rlink;
 }
 if(parent==NULL)
  return q;
 if(cur==parent->llink)
  parent->llink=q;
 else
  parent->rlink=q;
 freenode(cur);
 return root;
 }

void preorder(NODE root)
{
if(root!=NULL)
 {
  printf("%d  ",root->info);
  preorder(root->llink);
  preorder(root->rlink);
  }
 }
void postorder(NODE root)
{
if(root!=NULL)
 {

  postorder(root->llink);
  postorder(root->rlink);
  printf("%d  ",root->info);
  }
 }
void inorder(NODE root)
{
if(root!=NULL)
 {

  inorder(root->llink);
  printf("%d  ",root->info);
  inorder(root->rlink);
  }
 }
void main()
{
int item,choice;
NODE root=NULL;
for(;;)
{
printf("\n1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete
7.exit\n");
printf("enter the choice : ");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("enter the item : ");
            scanf("%d",&item);
            root=insert(root,item);
            break;
  case 2:
           if(root!=NULL)
            display(root,0);
            else
             printf("tree is empty \n");
            break;
  case 3:
           if(root!=NULL)
             preorder(root);
           else
             printf("tree is empty \n");
            break;
  case 4:
           if(root!=NULL)
             postorder(root);
           else
             printf("tree is empty \n");
            break;
  case 5:
           if(root!=NULL)
             inorder(root);
           else
             printf("tree is empty \n");
            break;
  case 6:printf("enter the item  : ");
            scanf("%d",&item);
            root=delete(root,item);
            break;
  default:exit(0);
             break;
        }
      }
 }
```



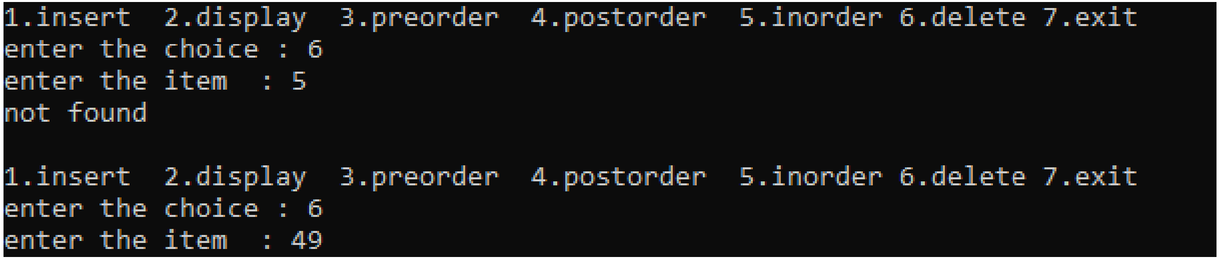
```
1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 2
tree is empty

1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 3
tree is empty

1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 4
tree is empty

1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 5
tree is empty

1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice :
```



```
1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 1
enter the item : 56

1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 2
  79
    56
49
    34
  23

1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 1
enter the item : 12

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 : 2
    100
  79
    56
49
    34
  23
    12
```



```
1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 2
    100
  79
    56
49
    34
  23
    12

1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 3
49  23  12  34  79  56  100
1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 3
49  23  12  34  79  56  100
1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 4
12  34  23  56  100  79  49
1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 5
12  23  34  49  56  79  100
1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
```



```
1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 6
enter the item  : 5
not found

1.insert  2.display  3.preorder  4.postorder  5.inorder 6.delete 7.exit
enter the choice : 6
enter the item  : 49
```

****