KIRAN M K

1BM19CS073

DATA STRUCTURES LAB RECORD

3rd Sem 'B' Section

Batch-1

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

Code:

```
#include <stdio.h>

#include<stdlib.h>

#define STACK_SIZE 5

int top=-1;

int s[5];

int item;

void push()

{

   if(top == STACK_SIZE - 1)

   {

      printf("STACK OVERFLOW, cannot push more elements into the stack\n");

      return;

   }

   else

   {

      top++;

      s[top] = item;

   }

}

int pop()

{

   if(top == -1)
```

```
    {
        printf("STACK UNDERFLOW, no elements in the stack\n");
        return 0;
    }
    else
    {
        return s[top--];
    }
}
void display()
{
    if(top == -1)
    {
        printf("STACK UNDERFLOW, no elements to display\n");
        return;
    }
    else
    {
        printf("The elements of the stack:\n");
        for(int i=0;i<=top;i++)
        printf("%d\n",s[i]);
    }
}
int main()
{
    int choice,item_deleted;

    for(;;)
```

```
    {
        printf("Enter\n1.push\n2.pop\n3.display\n4.exit\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("Enter the element to be inserted\n ");
                   scanf("%d",&item);
                   push();
                   break;
            case 2: {
                item_deleted = pop();
                if(item_deleted!=0)
                printf("Item popped = %d\n",item_deleted);
                break;
            }
            case 3: display();
                    break;
            default: exit(0);
        }
    }
}
```

Output:

```
Enter
1.push
2.pop
3.display
4.exit
1
Enter the element to be inserted
 30
Enter
1.push
2.pop
3.display
4.exit
1
Enter the element to be inserted
 40
Enter
1.push
2.pop
3.display
4.exit
1
Enter the element to be inserted
 50
Enter
1.push
2.pop
3.display
```

```
3.display
4.exit
1
Enter the element to be inserted
 60
Enter
1.push
2.pop
3.display
4.exit
1
Enter the element to be inserted
 70
Enter
1.push
2.pop
3.display
4.exit
1
Enter the element to be inserted
 78
STACK OVERFLOW, cannot push more elements into the stack
Enter
1.push
2.pop
3.display
4.exit
3
The elements of the stack:
30
```

```
30
40
50
60
70
Enter
1.push
2.pop
3.display
4.exit
2
Item popped = 70
Enter
1.push
2.pop
3.display
4.exit
2
Item popped = 60
Enter
1.push
2.pop
3.display
4.exit
2
Item popped = 50
Enter
1.push
2.pop
3.display
```

```
4.exit
2
Item popped = 40
Enter
1.push
2.pop
3.display
4.exit
2
Item popped = 30
Enter
1.push
2.pop
3.display
4.exit
2
STACK UNDERFLOW, no elements in the stack
Enter
1.push
2.pop
3.display
4.exit
3
STACK UNDERFLOW, no elements to display
Enter
1.push
2.pop
3.display
4.exit
4
```

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)

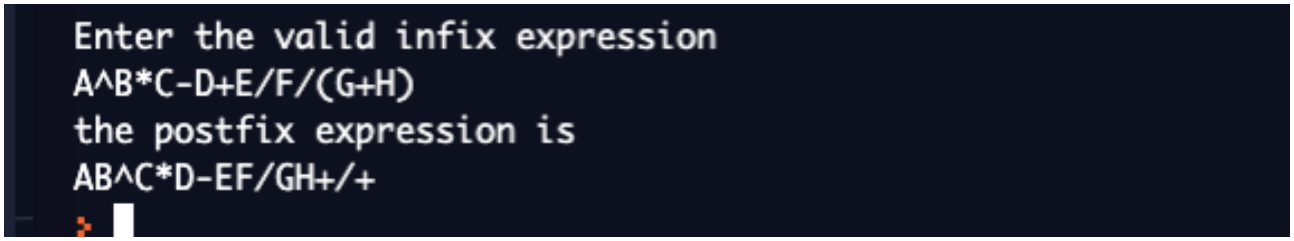
CODE:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int F(char symbol)
{
   switch(symbol)
   {
      case '+':
      case '-': return 2;
      case '*':
      case '/': return 4;
      case '^':
      case '$':return 5;
      case '(': return 0;
      case '#': return -1;
      default: return 8;
   }
}
int G(char symbol)
{
   switch(symbol)
   {
```

```
        case '+':
        case '-': return 1;
        case '*':
        case '/': return 3;
        case '^':
        case '$':return 6;
        case '(': return 9;
        case ')': return 0;
        default: return 7;
    }

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

```
        s[++top] = symbol;
        else
        top--;
    }
    while(s[top]!='#')
    {
        postfix[j++] = s[top--];
    }
    postfix[j] = '\0';
}
int main(void)
{
    char infix[20];
    char postfix[20];
    printf("Enter the valid infix expression\n");
    scanf("%s",infix);
    infix_postfix(infix,postfix);
    printf("the postfix expression is\n");
    printf("%s\n",postfix);
}
```

OUTPUT:



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

Code:

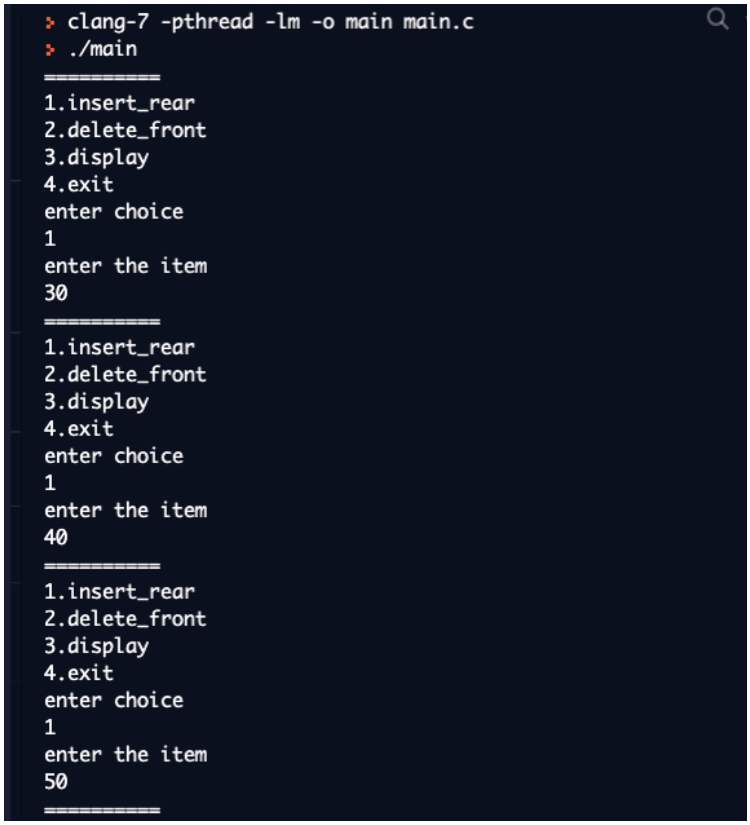
```
#include<stdio.h>

#include<stdlib.h>

#define qsize 5

int f=0,r=-1,ch;

int item,q[10];


int isfull()

  {

   return(r==qsize-1)?1:0;

  }

int isempty()

  {

   return(f>r)?1:0;

  }

void insert_rear()

  {

   if(isfull())

          {
```

```
            printf("==========queue overflow=========\n");
            return;
           }
   r=r+1;
   q[r]=item;
  }
void delete_front()
  {
   if(isempty())
           {
            printf("========queue empty=======\n");
            return;
           }
   printf("item deleted is %d\n",q[(f)++]);
   if(f>r)
           {
            f=0;
            r=-1;
           }
  }



void display()
  {
```

```
  int i;
  if(isempty())
         {
          printf("==========queue empty=======\n");
          return;
         }
  for(i=f;i<=r;i++)
        printf("%d\n",q[i]);
 }
int main()
 {


 for(;;)
  {
    printf("==========\n");
        printf("1.insert_rear\n2.delete_front\n3.display\n4.exit\n");
        printf("enter choice\n");
        scanf("%d",&ch);
        switch(ch)
         {
          case 1:printf("enter the item\n");
                        scanf("%d",&item);
                        insert_rear();
```

```
                    break;


            case 2:delete_front();
                    break;
            case 3:display();
                    break;
            default:exit(0);
           }
         }
}
```

OUTPUT:



```
3.display
4.exit
enter choice
3
30
40
50
25
98
==========
1.insert_rear
2.delete_front
3.display
4.exit
enter choice
2
item deleted is 30
==========
1.insert_rear
2.delete_front
3.display
4.exit
enter choice
2
item deleted is 40
==========
1.insert_rear
2.delete_front
3.display
4.exit
enter choice
2
item deleted is 50
==========
1.insert_rear
2.delete_front
3.display
4.exit
enter choice
2
item deleted is 25
==========
1.insert_rear
2.delete_front
3.display
4.exit
enter choice
2
item deleted is 98
==========
1.insert_rear
2.delete_front
3.display
4.exit
enter choice
2
========queue empty=======
==========
1.insert_rear
2.delete_front
========queue empty=======
==========
1.insert_rear
2.delete_front
3.display
4.exit
enter choice
4
```

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

Code:

```
#include<stdio.h>


#include<stdlib.h>

#define QUE_SIZE 3

int item,front=0,rear=-1,q[QUE_SIZE],count=0;

void insertrear()

{

if(count==QUE_SIZE)

{

printf("queue overflow\n");

return;

}

rear=(rear+1)%QUE_SIZE;

q[rear]=item;

count++;

}

int deletefront()

{

if(count==0) return -1;

item=q[front];

front=(front+1)%QUE_SIZE;
```

```
count=count-1;
return item;
}
void displayQ()
{
int i,f;
if(count==0)
{
printf("queue is empty\n");
return;
}
f=front;
printf("Contents of queue \n");
for(i=1;i<=count;i++)
{
printf("%d\n",q[f]);
f=(f+1)%QUE_SIZE;
}
}
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("item deleted =%d\n",item);
        break;
case 3:displayQ();
        break;
default:exit(0);
}
}


}
```

Output:

```
> clang-7 -pthread -lm -o main main.c
> ./main

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
23

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
34

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
56

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
90
queue overflow

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =23

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
78

1:insertrear
2:deletefront
```

```
3:display
4:exit
enter the choice
3
Contents of queue
34
56
78

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =34

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
3
Contents of queue
56
78

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
1
enter the item to be inserted
90

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
3
Contents of queue
56
78
90

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =56

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

```
2
item deleted =78

1:insertrear
2:deletefront
3:display
4:exit
enter the choice
2
item deleted =90

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

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

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

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.

Code:

```
#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 delete_front(NODE first)
{
    NODE temp;
    if(first == NULL)
    {
        printf("==================LIST EMPTY!Cannot delete==========\n");
        return first;
    }
    temp = first;
    temp = temp->link;
```

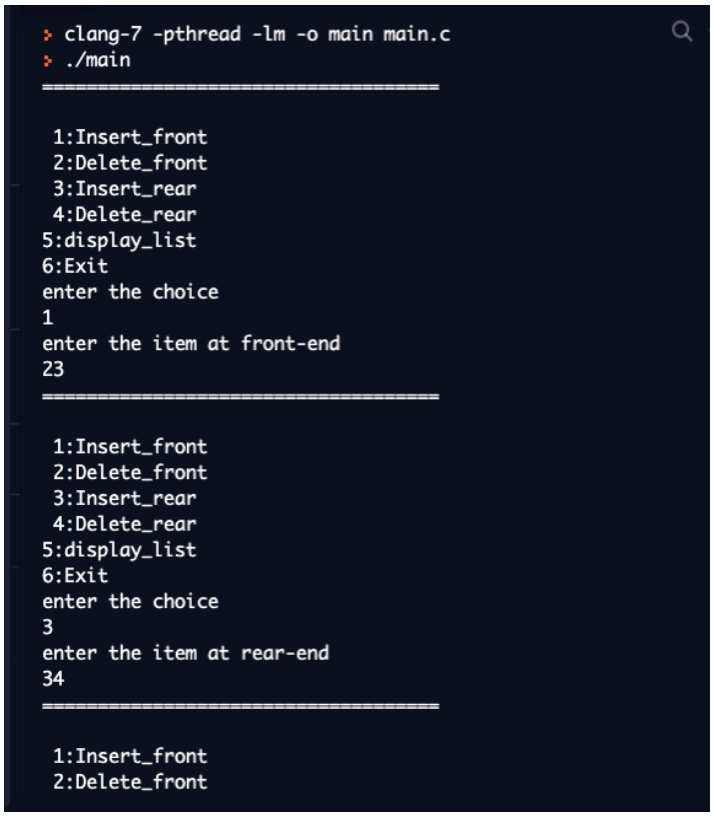
```
    printf("===========Item deleted at front end is = %d===========\n",first->info);
    free(first);
    return temp;

}
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 EMPTY!Cannot delete==========\n");
        return first;
```

```
    }
    if(first->link == NULL)
    {
        printf("==========Item deleted at rear end 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=======\n",cur->info);
        free(cur);
        prev->link = NULL;
        return first;
}

void display(NODE first)
{
    NODE temp;
    if(first == NULL )
    {
        printf("=================LIST EMPTY!Cannot display any item==========\n");
        return;
    }
```

```
    printf("================LIST 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");
        printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear
\n5:display_list\n6:Exit\n");
        printf("enter the choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("enter the item at front-end\n");
                    scanf("%d",&item);
                first=insert_front(first,item);
                break;
            case 2:first=delete_front(first);
                    break;
            case 3:printf("enter the item at rear-end\n");
                    scanf("%d",&item);
                first=insert_rear(first,item);
                    break;
```

```
        case 4:first=delete_rear(first);
                    break;
        case 5:display(first);
                    break;
        default:exit(0);
                   break;
      }
   }
   return 0;
}
```

OUTPUT:



```
 3:Insert_rear
 4:Delete_rear
5:display_list
6:Exit
enter the choice
1
enter the item at front-end
90
====================================

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
5:display_list
6:Exit
enter the choice
3
enter the item at rear-end
67
====================================

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

enter the item at front-end
45
====================================

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
5:display_list
6:Exit
enter the choice
3
enter the item at rear-end
78
====================================

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
5:display_list
6:Exit
enter the choice
5
================LIST ITEMS:::========
45
90
23
34
67
```

```
67
78
==================================

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
5:display_list
6:Exit
enter the choice
2
==========Item deleted at front end is = 45==========
==================================

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
5:display_list
6:Exit
enter the choice
2
==========Item deleted at front end is = 90==========
==================================

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
5:display_list
6:Exit
enter the choice
4
=========Item deleted at rear end is = 78=======
==================================

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
5:display_list
6:Exit
enter the choice
4
=========Item deleted at rear end is = 67=======
==================================

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
5:display_list
6:Exit
enter the choice
5
================LIST ITEMS:::=========
23
34
==================================
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
5:display_list
6:Exit
enter the choice
2
==========Item deleted at front end is = 23==========
==================================

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
5:display_list
6:Exit
enter the choice
4
=========Item deleted at rear end is = 34=======
==================================

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

==================LIST EMPTY!Cannot delete==========
==================================

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

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) Display the contents of the linked list.

Code:

```
#include<stdio.h>
#include<stdlib.h>
void display();
int count=0;
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 delete_front(NODE first)
{
   NODE temp;
   if(first == NULL)
   {
      printf("==================LIST EMPTY!Cannot delete==========\n");
      return first;
   }
   temp = first;
   temp = temp->link;
   printf("==========Item deleted at front end is = %d==========\n",first->info);
   freenode(first);
   count--;
   return temp;

}
NODE delete_rear(NODE first)
{

   NODE cur,prev;
   if(first == NULL)
   {
      printf("==================LIST EMPTY!Cannot delete==========\n");
```

```
        return first;
    }
    if(first->link == NULL)
    {
        printf("==========Item deleted at rear end is = %d=======\n",first->info);
        free(first);
        count--;
        return NULL;
    }
    prev = NULL;
    cur = first;
    while(cur->link != NULL)
    {
        prev = cur;
        cur = cur->link;
    }
    printf("==========Item deleted at rear end is = %d=======\n",cur->info);
        free(cur);
        prev->link = NULL;
        count--;
        return first;
}

NODE delete_pos(int pos,NODE first)
{
    NODE cur,prev;
    int count1;
    if(first == NULL)
```

```
{
   printf("==============LIST EMPTY!Cannot delete=========\n");
   return NULL;
}
if(pos<0)
{
   printf("===============Invalid position============\n");
   return NULL;
}
if(pos==1)
{
   cur = first;
   first = first->link;
   freenode(first);
   printf("=============Deletion successfull==========\n");
   return first;
}
prev = first;
cur=prev->link;
count1 = 1;
while(cur!=NULL && count1 != pos - 1 )
{
   prev = cur;
   cur = cur->link;
   count1++;
}
if(count1!=pos-1 && count<pos)
{
```

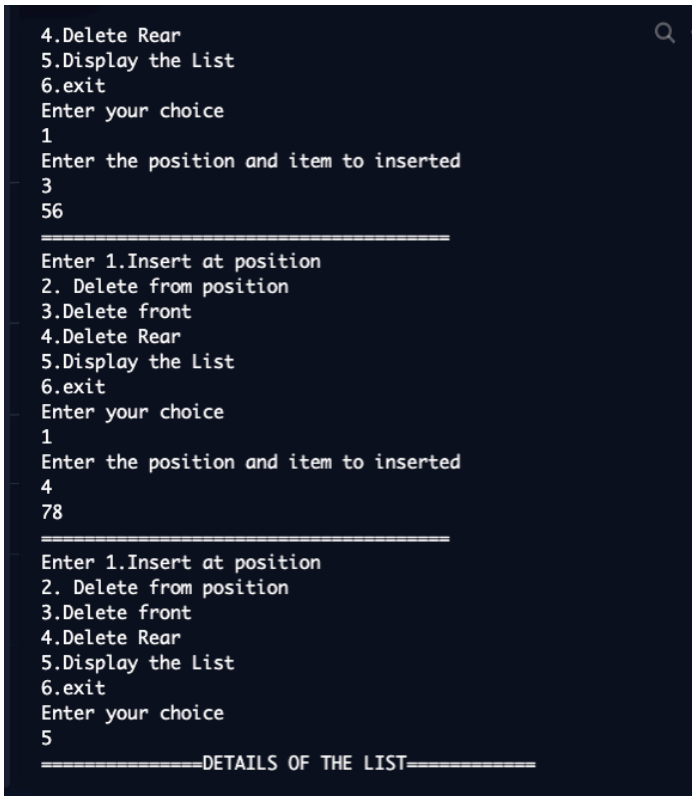
```
        printf("==========Invalid position==========\n ");
        return first;
    }
    printf("==========Deletion successfull==========\n");
    prev->link = cur->link;
    count--;
    freenode(cur);
    return first;
}
NODE insert_pos(NODE first,int key,int pos)
{
    int count1;
    NODE cur,temp,prev;
    temp = getnode();
    temp->info = key;
    temp->link = NULL;
    if(first == NULL && pos == 1)
    {
        count++;
        return temp;
    }
    if(first == NULL)
    {
        printf("==========INVALID POSITION==========\n");
        return NULL;
    }
    if(pos == 1)
    {
```

```
        temp->link = first;
        count++;
        return temp;
    }
    count1 = 1;
    prev = NULL;
    cur = first;
    while(cur!=NULL && count!=pos)
    {
        prev = cur;
        cur = cur->link;
        count1++;
    }
    if(count1 == pos)
    {
        prev->link = temp;
        temp->link = cur;
        count++;
        return first;
    }
    printf("================INVALID POSITION==============\n");
    return first;
}
void display(NODE first)
{
    NODE temp;
    if(first == NULL)
    {
```

```
        printf("============LIST EMPTY==============\n");
        return;
    }
    printf("===============DETAILS OF THE LIST============\n");

    for(temp = first;temp!=NULL;temp=temp->link)
    {
        printf("%d\n",temp->info);
    }
    return;
}
int main(void)
{
    NODE first=NULL;
    int item,pos,choice;
    for(;;)
    {
        printf("=====================================\n");
        printf("Enter 1.Insert at position\n2. Delete from position\n3.Delete front\n4.Delete
Rear\n5.Display the List\n6.exit\nEnter your choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: printf("Enter the position and item to inserted\n");
                    scanf("%d%d",&pos,&item);
                    first = insert_pos(first, item, pos);
                    break;
            case 2: printf("Enter the position to be deleted\n");
```

```
            scanf("%d",&pos);
            first = delete_pos(pos,first);
            break;
        case 3:first =  delete_front(first);
            break;
        case 4: first = delete_rear(first);
            break;
        case 5: display(first);
            break;
        default: exit(0);
      }
    }
}
```

OUTPUT:



```
23
90
56
78
=======================================
Enter 1.Insert at position
2. Delete from position
3.Delete front
4.Delete Rear
5.Display the List
6.exit
Enter your choice
2
Enter the position to be deleted
2
=============Deletion successfull==========
=======================================
Enter 1.Insert at position
2. Delete from position
3.Delete front
4.Delete Rear
5.Display the List
6.exit
Enter your choice
5
===============DETAILS OF THE LIST============
23
56
78
=======================================
```

```
Enter 1.Insert at position
2. Delete from position
3.Delete front
4.Delete Rear
5.Display the List
6.exit
Enter your choice
3
==========Item deleted at front end is = 23============
=======================================
Enter 1.Insert at position
2. Delete from position
3.Delete front
4.Delete Rear
5.Display the List
6.exit
Enter your choice
4
=========Item deleted at rear end is = 78=======
=======================================
Enter 1.Insert at position
2. Delete from position
3.Delete front
4.Delete Rear
5.Display the List
6.exit
Enter your choice
5
===============DETAILS OF THE LIST============
56
=======================================
Enter 1.Insert at position
2. Delete from position
3.Delete front
4.Delete Rear
5.Display the List
6.exit
Enter your choice
6
```

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

Code:

```
#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);
}
void display (NODE first)
{
   NODE temp;
   if(first==NULL)
   {
      printf("===============Linked List is empty ,Cannot Display items============\n");
      return;
   }
   printf("===========The contents of the linked list are: \n");
   for(temp=first;temp!=NULL;temp=temp->link)
   {
      printf("%d\n",temp->info);
   }
   printf("======================\n");
}



NODE concat(NODE first,NODE second)
{
   NODE cur;
   if(first==NULL)
      return second;
   if(second==NULL)
```

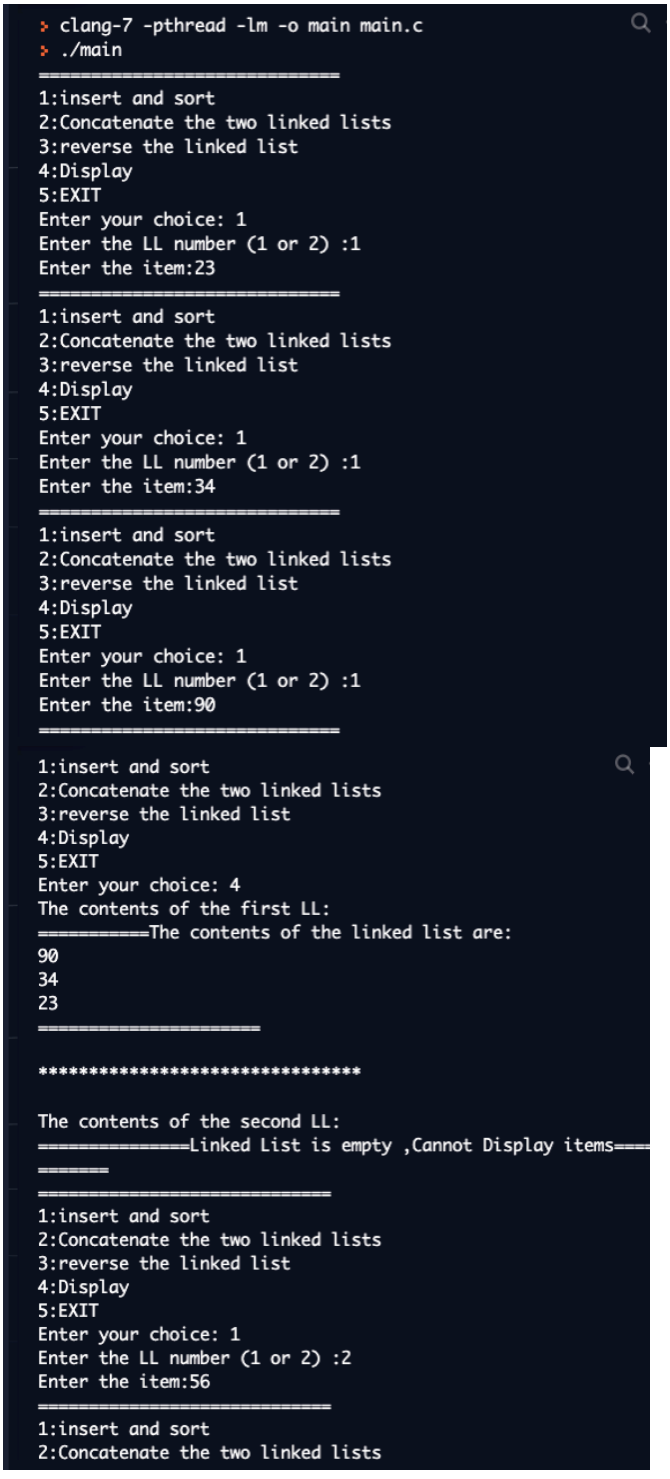
```
        return first;
    cur=first;
    while(cur->link!=NULL)
        cur=cur->link;
    cur->link=second;
    printf("=====Concatenated successfully======\n");
    return first;
}
NODE sort_and_insert(NODE first,int item)
{
    NODE temp,prev,cur;
    temp = getnode();
    temp->info = item;
    temp->link = NULL;
    if(first == NULL)
        return temp;
    if(item > first->info)//for ascending order , if(item < first->info)
    {
        temp->link = first;
        return temp;
    }
    prev = NULL;
    cur = first;
    while(cur!=NULL && item < cur->info)//for ascending order, item > cur->info
    {
        prev = cur;
        cur = cur->link;
    }
```

```
    prev->link = temp;
    temp->link = cur;
    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()
{
    NODE first=NULL;
    NODE second=NULL;
    int item, choice,llno;
    for(;;)
    {
        printf("============================\n");
        printf("1:insert and sort\n2:Concatenate the two linked lists\n3:reverse the linked
list\n4:Display\n5:EXIT\n");
```

```
printf("Enter your choice: ");
scanf("%d",&choice);
switch(choice)
{
   case 1: printf("Enter the LL number (1 or 2) :");
           scanf("%d",&llno);
           printf("Enter the item:");
           scanf("%d",&item);
           if(llno==1)
              first=sort_and_insert(first,item);
           else
              second=sort_and_insert(second,item);
           break;

   case 2: first=concat(first,second);
           break;
   case 3: printf("Enter the LL number (1 or 2) :");
           scanf("%d",&llno);
           if(llno==1)
              first=reverse(first);
           else
              second=reverse(second);
   case 4: printf("The contents of the first LL:\n");
           display(first);
           printf("\n*********************************\n");
           printf("\nThe contents of the second LL:\n");
           display(second);
           break;
```

```
        case 5: exit(0);
                break;
        default: printf("Enter a valid option\n");
      }
   }
   return 0;
}
```

OUTPUT:



```
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 1
Enter the LL number (1 or 2) :2
Enter the item:100
============================
1:insert and sort
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 2
=====Concatenated successfully======
============================
1:insert and sort
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 4
The contents of the first LL:
==========The contents of the linked list are:
90
34
23
100
56
====================

******************************

The contents of the second LL:
==========The contents of the linked list are:
100
56
====================
============================
1:insert and sort
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 3
Enter the LL number (1 or 2) :2
The contents of the first LL:
==========The contents of the linked list are:
90
34
23
100
====================

******************************

The contents of the second LL:
==========The contents of the linked list are:
56
100
```

```
====================
============================
1:insert and sort
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 4
The contents of the first LL:
==========The contents of the linked list are:
90
34
23
100
====================

******************************

The contents of the second LL:
==========The contents of the linked list are:
56
100
====================
============================
1:insert and sort
2:Concatenate the two linked lists
3:reverse the linked list
4:Display
5:EXIT
Enter your choice: 5
```

Program 8:

WAP to implement Stack & Queues using Linked Representation

Code:

Stack:

```
#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)
{
```

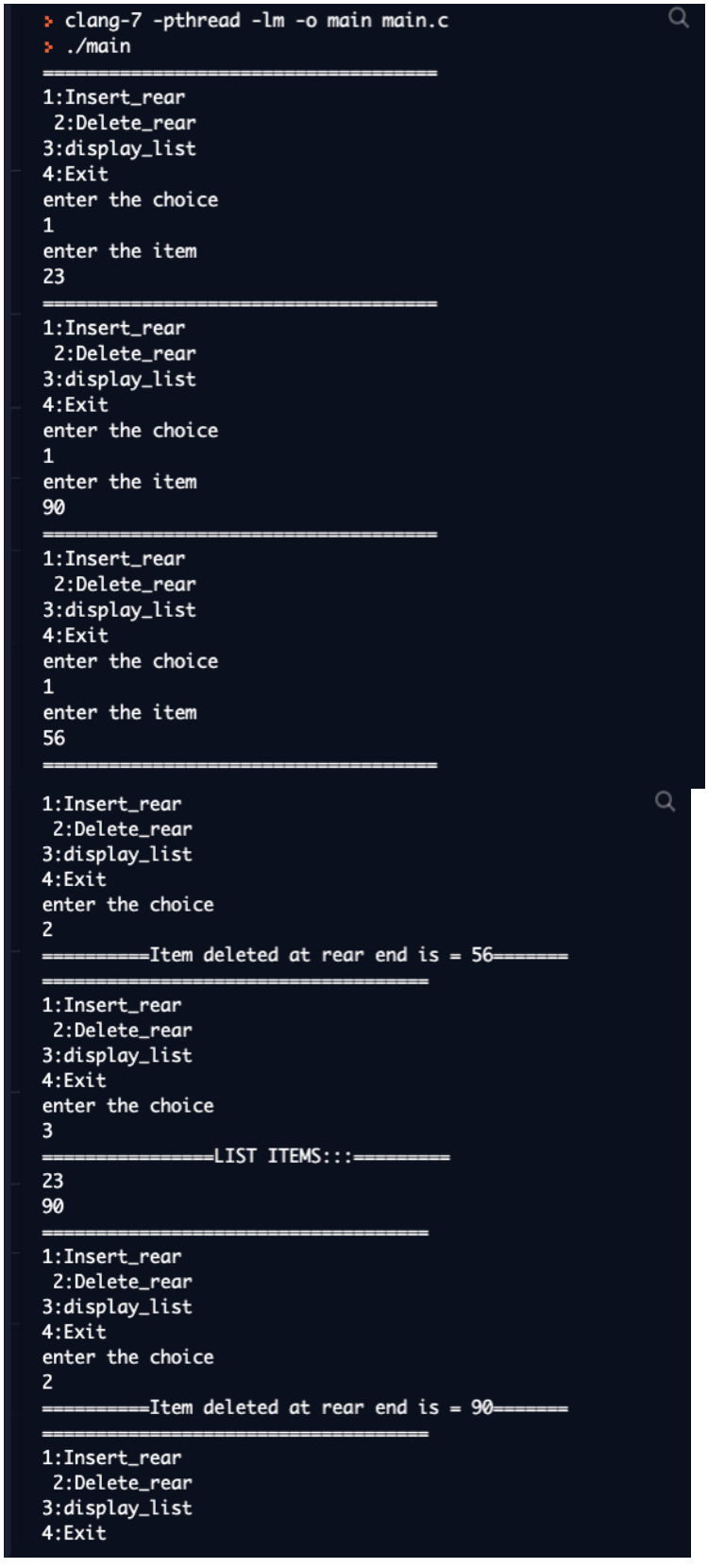
```
    free(x);
}



NODE insert_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 EMPTY!Cannot delete==========\n");
        return first;
    }
```

```
    if(first->link == NULL)
    {
        printf("==========Item deleted at rear end 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=======\n",cur->info);
        free(cur);
        prev->link = NULL;
        return first;
}

void display(NODE first)
{
    NODE temp;
    if(first == NULL )
    {
        printf("==================LIST EMPTY!Cannot display any item==========\n");
        return;
    }
    printf("================LIST 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");
        printf("1:Insert_rear\n 2:Delete_rear              \n3:display_list\n4:Exit\n");
        printf("enter the choice\n");
        scanf("%d",&choice);
        switch(choice)
        {

            case 1:printf("enter the item\n");
                        scanf("%d",&item);
                    first=insert_rear(first,item);
                        break;
            case 2:first=delete_rear(first);
                        break;
            case 3:display(first);
                        break;
            default:exit(0);
                       break;
        }
```

```
    }
    return 0;
}
```

OUTPUT:



Queue:

```
#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 delete_front(NODE first)
{
    NODE temp;
    if(first == NULL)
    {
        printf("==================LIST 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;
}


void display(NODE first)
{
    NODE temp;
    if(first == NULL )
    {
        printf("==================LIST EMPTY!Cannot display any item==========\n");
        return;
    }
    printf("================LIST 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");
        printf("\n1:Delete_front\n2:Insert_rear\n3:display_list\n4:Exit\n");
        printf("enter the choice\n");
        scanf("%d",&choice);
```

```
    switch(choice)
    {

       case 1:first=delete_front(first);
                    break;
       case 2:printf("enter the item at rear-end\n");
                    scanf("%d",&item);
                first=insert_rear(first,item);
                    break;

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

Output:

```
clang-7 -pthread -lm -o main main.c
./main
=================================

1:Delete_front
2:Insert_rear
3:display_list
4:Exit
enter the choice
1
================LIST EMPTY!Cannot delete==========
=================================

1:Delete_front
2:Insert_rear
3:display_list
4:Exit
enter the choice
2
enter the item at rear-end
23
=================================

1:Delete_front
2:Insert_rear
3:display_list
4:Exit
enter the choice
2
enter the item at rear-end
```

```
78
==================================

1:Delete_front
2:Insert_rear
3:display_list
4:Exit
enter the choice
2
enter the item at rear-end
45
==================================

1:Delete_front
2:Insert_rear
3:display_list
4:Exit
enter the choice
3
================LIST ITEMS:::=========
23
78
45
==================================

1:Delete_front
2:Insert_rear
3:display_list
4:Exit
enter the choice
1
==========Item deleted at front end is = 23==========
==================================

1:Delete_front
2:Insert_rear
3:display_list
4:Exit
enter the choice
1
==========Item deleted at front end is = 78==========
==================================

1:Delete_front
2:Insert_rear
3:display_list
4:Exit
enter the choice
4
```

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

Program:

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

struct node
{
   int info;
   struct node *llink;
   struct node *rlink;
};
typedef struct node *NODE;

NODE getnode()
{
   NODE x;
   x=(NODE)malloc (sizeof(struct node));
   if(x==NULL)
   {
      printf("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;
    cur=head->rlink;
    head->rlink=temp;
    temp->llink=head;
    temp->rlink=cur;
    cur->llink=temp;
    return head;
}
NODE dinsert_rear(int item,NODE head)
{
    NODE temp,cur;
    temp=getnode();
    temp->info=item;
    cur=head->llink;
    head->llink=temp;
    temp->rlink=head;
```

```
    temp->llink=cur;
    cur->rlink=temp;
    return head;
}


NODE insert_leftpos(int item,NODE head)
{
    NODE temp,cur,prev;
    int item2;
    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;
    temp=getnode();
```

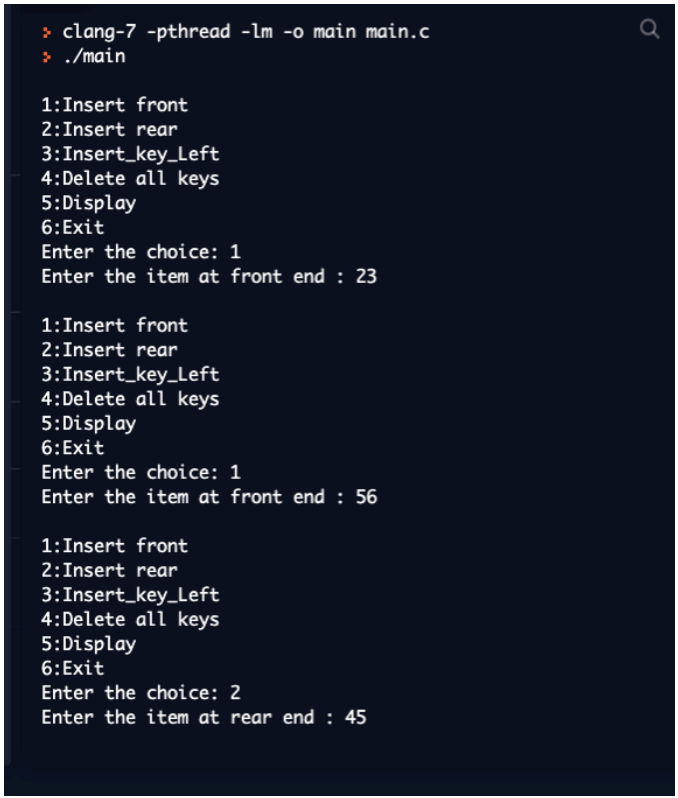
```
    printf("Enter towards left of %d : ",item);
    scanf("%d",&item2);
    temp->info=item2;
    prev->rlink=temp;
    temp->llink=prev;
    cur->llink=temp;
    temp->rlink=cur;
    return head;
}

NODE delete_all_key(int item, NODE head)
{
    NODE prev,cur,next;
    int count;
    if(head->rlink==head)
    {
        printf("List empty\n");
        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\n");
    else
        printf("Key found at %d positions !! and are deleted",count);
    return head;

}



void display(NODE head)
{
    NODE temp;
    if(head->rlink==head)
    {
        printf("The DLL is empty");
        return;
    }
    printf("the contents of the DLL\n");
    temp=head->rlink;
```

```
    while(temp!=head)
    {
        printf("%d\n",temp->info);
        temp=temp->rlink;
    }
    printf("\n");
}

int main()
{
    NODE head,last;
    int item,choice;
    head=getnode();
    head->rlink=head;
    head->llink=head;
    for(;;)
    {
        printf("\n1:Insert front\n2:Insert rear\n3:Insert_key_Left\n4:Delete all
            keys\n5:Display\n6:Exit\n");
        printf("Enter the choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
                case 1: printf("Enter the item at front end : ");
                        scanf("%d",&item);
                        last=dinsert_front(item,head);
                        break;
                case 2: printf("Enter the item at rear end : ");
```

```
                scanf("%d",&item);
                last=dinsert_rear(item,head);
                break;

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

OUTPUT:



```
1:Insert front
2:Insert rear
3:Insert_key_Left
4:Delete all keys
5:Display
6:Exit
Enter the choice: 3
Enter the key item: 56
Enter towards left of 56 : 90

1:Insert front
2:Insert rear
3:Insert_key_Left
4:Delete all keys
5:Display
6:Exit
Enter the choice: 5
the contents of the DLL
90
56
23
45


1:Insert front
2:Insert rear
3:Insert_key_Left
4:Delete all keys
5:Display
6:Exit
Enter the choice: 4
Enter the key item: 23
Key found at 1 positions !! and are deleted
1:Insert front
2:Insert rear
3:Insert_key_Left
4:Delete all keys
5:Display
6:Exit
Enter the choice: 5
the contents of the DLL
90
56
45


1:Insert front
2:Insert rear
3:Insert_key_Left
4:Delete all keys
5:Display
6:Exit
Enter the choice: 6
```

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

Code:

```
#include<stdio.h>

#include<stdlib.h>


struct node

{

   int info;

   struct node *llink;

   struct node *rlink;

};

typedef struct node *NODE;


NODE getnode()

{

   NODE x;

   x = (NODE)malloc(sizeof(struct node));

   if(x == NULL)

   {

      printf("Mmeory 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->info = item;
   temp->llink = NULL;
   temp->rlink = NULL;
   if(root == NULL)
      return temp;
   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("================\nTREE EMPTY\n============\n");
      return root;
   }
   parent=NULL;
   cur=root;
   while(cur!=NULL&&item!=cur->info)
   {
      parent=cur;
      cur=(item<cur->info)?cur->llink:cur->rlink;
   }
```

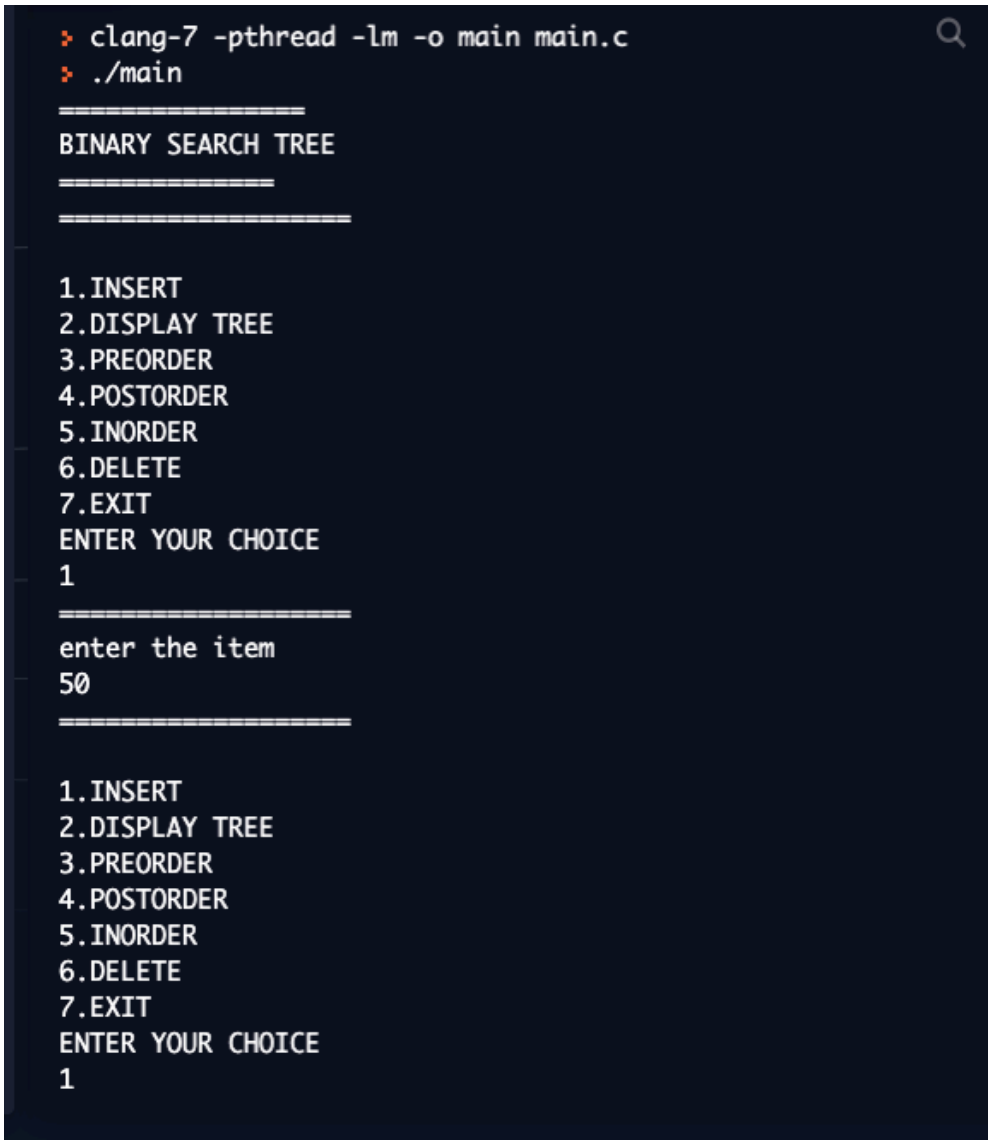
```
    if(cur==NULL)
    {
        printf("not found\n");
        return root;
    }
    if(cur->llink==NULL)
        q=cur->rlink;
    else if(cur->rlink==NULL)
        q=cur->llink;
    else
    {
        suc=cur->rlink;
        while(suc->llink!=NULL)
            suc=suc->llink;
        suc->llink=cur->llink;
        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(void)
```

```
{
   int item,choice;
   NODE root=NULL;
   printf("================\nBINARY SEARCH TREE\n=============\n");
   for(;;)
   {
      printf("====================\n");
      printf("\n1.INSERT \n2.DISPLAY
TREE\n3.PREORDER\n4.POSTORDER\n5.INORDER\n6.DELETE\n7.EXIT\n");
      printf("ENTER YOUR CHOICE\n");

      scanf("%d",&choice);
      printf("====================\n");
      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);

          }
        }
}
```

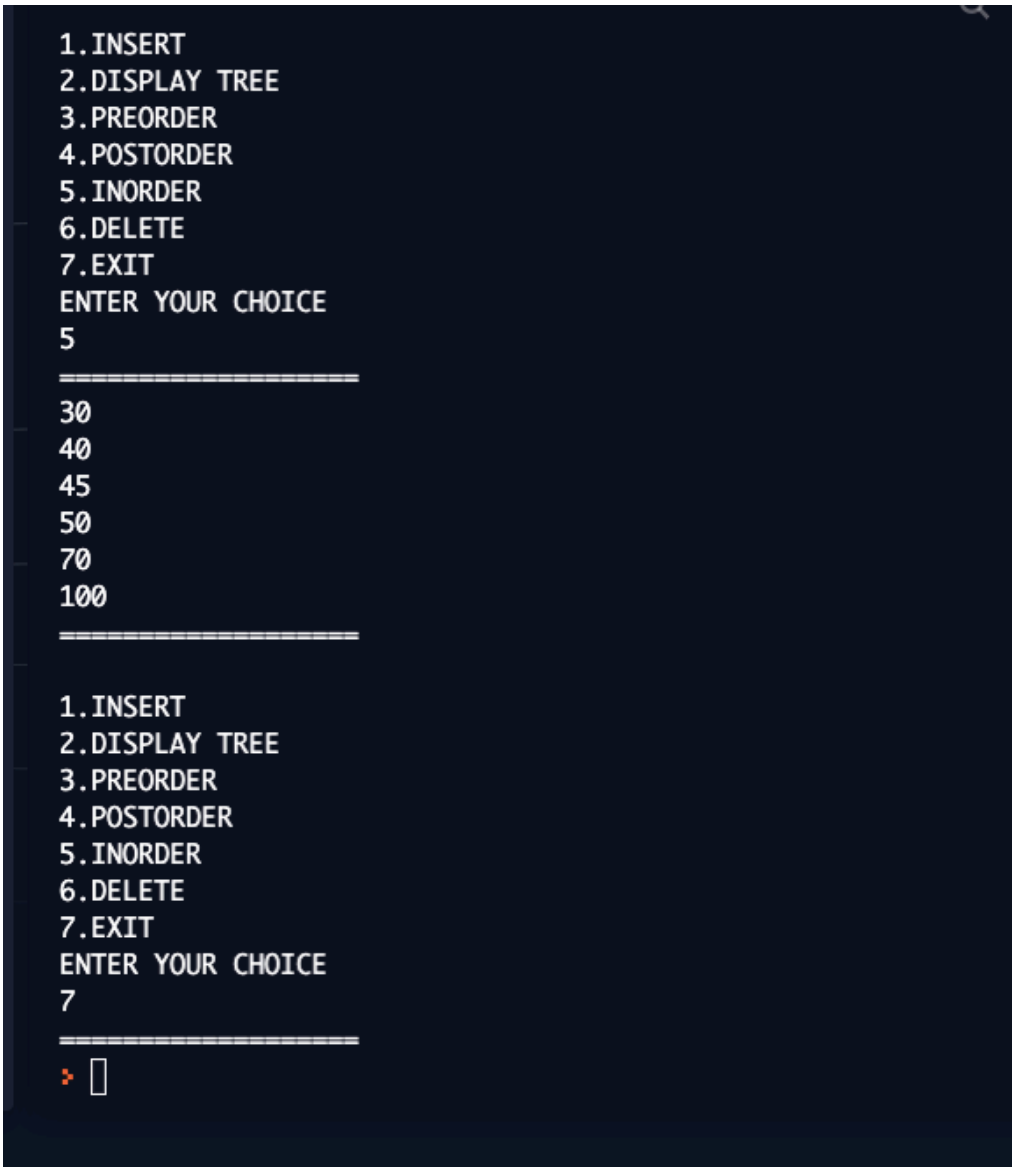
OUTPUT:



```
==================
enter the item
70
==================

1.INSERT
2.DISPLAY TREE
3.PREORDER
4.POSTORDER
5.INORDER
6.DELETE
7.EXIT
ENTER YOUR CHOICE
1
==================
enter the item
40
==================

1.INSERT
2.DISPLAY TREE
3.PREORDER
4.POSTORDER
5.INORDER
6.DELETE
7.EXIT
ENTER YOUR CHOICE
1
==================
enter the item
```

```
100
==================

1.INSERT
2.DISPLAY TREE
3.PREORDER
4.POSTORDER
5.INORDER
6.DELETE
7.EXIT
ENTER YOUR CHOICE
1
==================
enter the item
30
==================

1.INSERT
2.DISPLAY TREE
3.PREORDER
4.POSTORDER
5.INORDER
6.DELETE
7.EXIT
ENTER YOUR CHOICE
1
==================
enter the item
45
==================
```

```
1.INSERT
2.DISPLAY TREE
3.PREORDER
4.POSTORDER
5.INORDER
6.DELETE
7.EXIT
ENTER YOUR CHOICE
2
==================
            100
      70
50
            45
      40
            30
==================

1.INSERT
2.DISPLAY TREE
3.PREORDER
4.POSTORDER
5.INORDER
6.DELETE
7.EXIT
ENTER YOUR CHOICE
3
==================
50
40
30
45
70
100
==================

1.INSERT
2.DISPLAY TREE
3.PREORDER
4.POSTORDER
5.INORDER
6.DELETE
7.EXIT
ENTER YOUR CHOICE
4
==================
30
45
40
100
70
50
==================

1.INSERT
2.DISPLAY TREE
3.PREORDER
4.POSTORDER
```


1.INSERT
2.DISPLAY TREE
3.PREORDER
4.POSTORDER
5.INORDER
6.DELETE
7.EXIT
ENTER YOUR CHOICE
5
==========
30
40
45
50
70
100
==========
1.INSERT
2.DISPLAY TREE
3.PREORDER
4.POSTORDER
5.INORDER
6.DELETE
7.EXIT
ENTER YOUR CHOICE
7
==========