```c
/*
  TITLE:IMPLEMENTATION OF DOUBLY LINKED LIST
  NAME:Tauseef Mushtaque Ali Shaikh
  CLASS: S.Y.[CO]
  ROLLNO: 18CO63
  SUBJECT: DS
  DATE: 23/9/19
  DISCRIPTION: In this Program a doubly linked list is created and different function is performed i.e. insert, remove, display.
*/
#include<stdio.h>
#include<stdlib.h>

struct DLL
{
int data;
struct DLL *next, *prev;
};

struct DLL *insertAtEnd(struct DLL *h,int d)
{
    struct DLL *p,*tmp;
    p=(struct DLL *)malloc(sizeof(struct DLL));
    p->data=d;
    p->next=NULL;
    p->prev=NULL;
    if(h==NULL)
    {
        h=p;
    }
else
{
tmp=h;
while(tmp->next!=NULL)
tmp=tmp->next;
tmp->next=p;
p->prev=tmp;
}
return h;
}

struct DLL *insertAtStart(struct DLL *h,int d)
{
struct DLL *p;
p=(struct DLL *)malloc(sizeof(struct DLL));
p->data=d;
p->next=h;
p->prev=NULL;
if(h!=NULL)
h->prev=p;
h=p;
return h;
}

struct DLL *insertAfter(struct DLL *h,int key,int d)
{
struct DLL *p,*tmp;
p=(struct DLL *)malloc(sizeof(struct DLL));
p->data=d;
p->next=NULL;
p->prev=NULL;
if(h==NULL)
{
h=p;
```

```
64  }
65  else
66  {
67  tmp=h;
68  while(tmp!=NULL && tmp->data!=key )
69  tmp=tmp->next;
70  if(tmp!=NULL)
71  {
72  p->next=tmp->next;
73  p->prev=tmp;
74  if(tmp->next!=NULL)
75  (tmp->next)->prev=p;
76  tmp->next=p;
77  }
78  else
79  printf("\n\tGiven Node %d does not exist in the Linked List.",key);
80  }
81  return h;
82  }
83
84  struct DLL *removelast(struct DLL *h)
85  {
86  struct DLL *tmp;
87  tmp=h;
88  if(h!=NULL)
89  {
90  if(h->next!=NULL)
91  {
92  while(tmp->next!=NULL)
93  tmp=tmp->next;
94  (tmp->prev)->next=NULL;
95  }
96  else
97  h=NULL;
98  free(tmp);
99  }
100 else
101 printf("\nLL is empty.");
102 return h;
103 }
104
105 struct DLL *removeAfter(struct DLL *h,int key)
106 {
107 struct DLL *tmp,*p;
108 tmp=h;
109 if(h!=NULL)
110 {
111 while(tmp!=NULL && tmp->data!=key)
112 tmp=tmp->next;
113 if(tmp!=NULL)
114 {
115 if(tmp->next!=NULL)
116 {
117 p=tmp->next;
118 if(p->next!=NULL)
119 (p->next)->prev=tmp;
120 tmp->next=p->next;
121 p->next=NULL;
122 p->prev=NULL;
123 free(p);
124 }
125 else
126 printf("\nGiven Node is the last Node.");
127 }
```

```
else
printf("\nGiven key does not exist.");
}
else
printf("\nLL is empty.");
return h;
}

void display(struct DLL *h)
{
struct DLL *tmp;
tmp=h;
if(h!=NULL)
{
printf("\n\n\t\tLinked List Contents..\n");
while(tmp!=NULL)
{
printf("\t%d\n",tmp->data);
tmp=tmp->next;
}
}
else
printf("\nLL is empty.");
}

void displayRev(struct DLL *h)
{
struct DLL *tmp;
tmp=h;
if(h!=NULL)
{
printf("\n\n\t\tLinked List Contents in Reverse Order..\n");
while(tmp->next!=NULL)
tmp=tmp->next;
do
{
printf("\t%d\n",tmp->data);
tmp=tmp->prev;
}while(tmp!=NULL);
}
else
printf("\nLL is empty.");
}

int main()
{
            struct DLL *head;
            int ch,d,k;
            head=NULL;
            while(1)
        {

            printf("\n\t\tMENU");
            printf("\n\n1. INSERT AT END\n2. INSERT AFTER\n3. INSERT AT START\n4.
            REMOVE FROM LAST\n5. REMOVE AFTER\n6. DISPLAY\n7. DISPLAY REVERSE\n0.
            EXIT\n");
            printf("\nENTER THE CHOICE: ");
            scanf("%d",&ch);

            switch(ch)
            {
                case 1:
                printf("\nEnter the element to be insert: ");
                scanf("%d",&d);
```

```
                    head=insertAtEnd(head,d);
                    break;

                    case 2:
                    printf("Enter the KEY: ");
                    scanf("%d",&k);
                    printf("\nEnter the element to be insert: ");
                    scanf("%d",&d);
                    head=insertAfter(head,k,d);
                    break;

                    case 3:
                    printf("\nEnter the element to be insert: ");
                    scanf("%d",&d);
                    head=insertAtStart(head,d);
                    break;

                    case 4:
                    head=removelast(head);
                    break;

                    case 5:
                    printf("Enter the KEY: ");
                    scanf("%d",&k);
                    head=removeAfter(head,k);
                    break;

                    case 6:
                    display(head);
                    break;

                    case 7:
                    displayRev(head);

                    case 0:
                    exit(0);
                    break;

                    default:
                    printf("INVALID CHOICE!!");

                }


}
        return 0;
                }
```