```c
/*
  TITLE:IMPLEMENTATION OF SINGLY LINKED LIST
  NAME:Tauseef Mushtaque Ali Shaikh
  CLASS: S.Y.[CO]
  ROLLNO: 18CO63
  SUBJECT: DS
  DATE: 16/9/19
  DISCRIPTION: In this Program a singly linked list is created and different
  function is performed i.e. insert, remove, display.
*/
#include<stdio.h>
#include<stdlib.h>
struct sll
{
int data;
struct sll *next;
};
struct sll *einsert(struct sll *h, int d)
{
    struct sll *p, *temp;
    p=(struct sll*)malloc(sizeof(struct sll));
    if(p==NULL)
    {
        printf("\n NOT ENOUGH SPACE!\nNODE CANNOT BE INSERTED\n");
        return h;
        }
        p->data=d;
        p->next=NULL;
        if(h==NULL)
        {
            h=p;
            }
        else
        {
            temp=h;
            while(temp->next!=NULL)
            temp=temp->next;
            temp->next=p;
            }
    return h;
    }

struct sll *sinsert(struct sll *h, int d)
{
    struct sll*p;
    p=(struct sll*)malloc(sizeof(struct sll));
    p->data=d;
    p->next=h;
    h=p;
    return h;
    }

struct sll *ainsert(struct sll *h, int key, int d)
{
    struct sll *p, *temp;
    p=(struct sll*)malloc(sizeof(struct sll));
    p->data=d;
    p->next=NULL;
    if(h==NULL)
    {
        h=p;
        }
    else
    {
```

```
64          temp=h;
65          while(temp!=NULL && temp->data!=key)
66          temp=temp->next;
67          if(temp!=NULL)
68          {
69              p->next=temp->next;
70              temp->next=p;
71              }
72              else
73              {
74                  printf("\nGIVEN NODE %d DOES NOT EXIST IN THE LINKED LIST!", key);
75                  free(p);
76                  }
77          }
78          return h;
79      }
80
81  struct sll *lremove(struct sll *h)
82  {
83      struct sll *temp, *prev;
84      temp=h;
85      if(h!=NULL)
86      {
87          if(h->next!=NULL)
88          {
89              while(temp->next!=NULL)
90              {
91                  prev=temp;
92                  temp=temp->next;
93              }
94              prev->next=NULL;
95          }
96          else
97          {
98          h=NULL;
99          free(temp);
100         }
101     }
102         else
103         {
104         printf("\nLINK LIST IS EMPTY!");
105         return h;
106         }
107     return h;
108 }
109
110 struct sll *fremove(struct sll *h)
111 {
112     struct sll *temp;
113     temp=h;
114     if(h!=NULL)
115     {
116         h=h->next;
117         free(temp);
118         }
119     else
120     {
121         printf("\nLINK LIST IS EMPTY!");
122     }
123         return h;
124     }
125
126 struct sll *aremove(struct all*h, int key)
127 {
```

```
128     struct sll *temp, *p;
129     temp=h;
130     if(h!=NULL)
131     {
132     while(temp!=NULL && temp->data!=key)
133     {
134         temp=temp->next;
135         }
136     if(temp!=NULL)
137     {
138     if(temp->next!=NULL)
139     {
140     p=temp->next;
141     temp->next=temp->next;
142     p->next=NULL;
143     free(p);
144         }
145     else
146     {
147         printf("\nGIVEN NODE IS THE LAST NODE!");
148         }
149         }
150     else
151     {
152         printf("\nGIVEN KEY DOES NOT EXIST!");
153         }
154     }
155     else
156     {
157         printf("\nLINKED LIST IS EMPTY!");
158         }
159     return h;
160 }
161
162 void display(struct sll *h)
163 {
164     struct sll *temp;
165     temp=h;
166     if(h!=NULL)
167     {
168         printf("\nLINKED LIST CONTENTS ARE:\n");
169         while(temp!=NULL)
170         {
171             printf("\n%d\t", temp->data);
172             temp=temp->next;
173             }
174         }
175         else
176         {
177             printf("\nLINKED LIST IS EMPTY!");
178             }
179     }
180
181 int main()
182 {
183     struct sll *head;
184     int ch,d,key;
185     head=NULL;
186     while(1)
187     {
188     printf("\n\tMENU\n1.INSERT AT START\n2.INSERT AT END\n3.INSERT AFTER\n4.REMOVE
    FROM START\n5.REMOVE FROM END\n6.REMOVE AFTER\n7.DISPLAY\n0.EXIT");
189     printf("\nEnter your choice::\n");
190     scanf("%d",&ch);
```

```
    switch(ch)
    {
        case 1:
        {
            printf("\nENTER DATA: ");
            scanf("%d",&d);
            head=sinsert(head,d);
            break;
            }
        case 2:
        {
            printf("\nENTER DATA: ");
            scanf("%d",&d);
            head=einsert(head,d);
            break;
            }
        case 3:
        {
            printf("\nENTER DATA: ");
            scanf("%d",&d);
            printf("\nENTER KEY: ");
            scanf("%d",&key);
            head=ainsert(head,key,d);
            break;
            }
            case 4:
        {
            head=fremove(head);
            break;
            }
            case 5:
        {
            head=lremove(head);
            break;
            }
        case 6:
        {
            printf("\nENTER KEY: ");
            scanf("%d",&d);
            head=aremove(head,d);
            break;
            }
        case 7:
        {
            display(head);
            }
        case 0:
        {
        exit(0);
            }
        default:
        {
            printf("\nCHOOSE A VALID OPTION!");
            }
        }
    }
    }
```