```c
//Singly Linked List Operations
#include<stdio.h>
#include<stdlib.h>

struct node
{
    int data;
    struct node *link;
};

struct node *head;

void Display()
{
    struct node *ptr;
    if(head==NULL)
        printf("\nLinked list is Empty... ");
```

```c
void main()
{
    int opt,x,key;
    do
    {
        printf("\nEnter the option\n1.Insert at Front\n2.Insert
        scanf("%d",&opt);
        switch(opt)
        {
            case 1: printf("Enter the new data: ");
                    scanf("%d",&x);
                    Insert_Front(x);
                    break;
            case 2: printf("Enter the new data: ");
                    scanf("%d",&x);
                    Insert_End(x);
                    break;
```

Singly Linked List.c

```
167                     break;
168                 case 2: printf("Enter the new data: ");
169                     scanf("%d",&x);
170                     Insert_End(x);
171                     break;
172                 case 3: printf("Enter the search key: ");
173                     scanf("%d",&key);
174                     printf("Enter the new data: ");
175                     scanf("%d",&x);
176                     Insert_After(key,x);
177                     break;
178                 case 4: Delete_Front();
179                     break;
180                 case 5: Delete_End();
181                     break;
182                 case 6: printf("Enter the node to be deleted: ");
183                     scanf("%d",&key);
```

```
                scanf("%d",&key);
                printf("Enter the new data: ");
                scanf("%d",&x);
                Insert_After(key,x);
                break;
            case 4: Delete_Front();
                break;
            case 5: Delete_End();
                break;
            case 6: printf("Enter the node to be deleted: ");
                scanf("%d",&key);
                Delete_Any(key);
                break;
            case 7: Display();
                break;
        }
    }while(opt!=8);
```

```c
void Display()
{
    struct node *ptr;
    if(head==NULL)
        printf("\nLinked list is Empty...");
    else
    {
        ptr=head;
        printf("\nLinked list elements are: ");
        while(ptr!=NULL)
        {
            printf("%d\t",ptr->data);
            ptr=ptr->link;
        }
    }
}
```

Singly Linked List.c

```
27      }
28  }
29
30  void Insert_Front(int x)
31  {
32      struct node *new;
33      new=(struct node *)malloc(sizeof(struct node ));
34      new->data=x;
35      new->link=head;
36      head=new;
37      Display();
38  }
39
40  void Insert_End(int x)
41  {
42      struct node *new,*ptr;
43      new=(struct node *)malloc(sizeof(struct node ));
```

public void *__cdecl malloc (size_t _Size)

Line: 33 Col: 50 Sel: 19 Lines: 191 Length: 3303 Insert Done parsing in 0.031 seconds

Singly Linked List.c

```
40 void Insert_End(int x)
41 {
42     struct node *new,*ptr;
43     new=(struct node *)malloc(sizeof(struct node ));
44     new->data=x;
45     new->link=NULL;
46     if(head==NULL)
47         head=new;
48     else
49     {
50         ptr=head;
51         while(ptr->link!=NULL)
52             ptr=ptr->link;
53         ptr->link=new;
54     }
55     Display();
56 }
```

Line: 40 Col: 11 Sel: 0 Lines: 191 Length: 3303 Insert Done parsing in 0.031 seconds

Singly Linked List.c

```
58 void Insert_After(key,x)
59 {
60     struct node *new,*ptr;
61     if(head==NULL)
62         printf("Search key not found. Insertion is not possible.
63     else
64     {
65         ptr=head;
66         while(ptr->data!=key && ptr->link!=NULL)
67             ptr=ptr->link;
68         if(ptr->data!=key)
69             printf("Search data not found. Insertion not possibl
70         else
71         {
72             new=(struct node *)malloc(sizeof(struct node ));
73             new->data=x;
74             new->link=ptr->link;
```

Line: 63 Col: 6 Sel: 0 Lines: 191 Length: 3303 Insert Done parsing in 0.031 seconds

```
62          printf("Search key not found. Insertion is not possible.
63      else
64      {
65          ptr=head;
66          while(ptr->data!=key && ptr->link!=NULL)
67              ptr=ptr->link;
68          if(ptr->data!=key)
69              printf("Search data not found. Insertion not possibl
70          else
71          {
72              new=(struct node *)malloc(sizeof(struct node ));
73              new->data=x;
74              new->link=ptr->link;
75              ptr->link=new;
76          }
77      }
78
```

```
80  }
81
82  void Delete_Front()
83  {
84      struct node *temp;
85      if(head==NULL)
86          printf("List is Empty. Deletion not possible..");
87      else
88      {
89          temp=head;
90          head=head->link;
91          free(temp);
92      }
93
94      Display();
95  }
96
```

Linked List.c

```
96
97 void Delete_End()
98 {
99      struct node *prev,*curr,*temp;
100     if(head==NULL)
101         printf("List is Empty. Deletion not possible..");
102     else if(head->link==NULL)
103     {
104         temp=head;
105         head=NULL;
106         free(temp);
107     }
108     else
109     {
110         prev=head;
111         curr=head->link;
112         while(curr->link!=NULL)
```

Line: 100 Col: 11 Sel: 6 Lines: 191 Length: 3303 Insert Done parsing in 0.031 seconds

Singly Linked List.c

```
105             head=NULL;
106             free(temp);
107         }
108         else
109         {
110             prev=head;
111             curr=head->link;
112             while(curr->link!=NULL)
113             {
114                 prev=curr;
115                 curr=curr->link;
116             }
117             prev->link=NULL;
118             free(curr);
119         }
120         Display();
121 }
```

```
123 void Delete_Any(int key)
124 {
125     struct node *prev,*curr,*temp;
126     if(head==NULL)
127         printf("List is Empty. Deletion not possible..");
128     else if(head->data==key)
129     {
130         temp=head;
131         head=head->link;
132         free(temp);
133     }
134     else
135     {
136         prev=head;
137         curr=head;
138         while(curr->data!=key && curr->link!=NULL)
139         {
```

```
134     else
135     {
136         prev=head;
137         curr=head;
138         while(curr->data!=key && curr->link!=NULL)
139         {
140             prev=curr;
141             curr=curr->link;
142         }
143         if(curr->data!=key)
144             printf("Search data not found. Deletion is not possi
145         else
146         {
147             prev->link=curr->link;
148             free(curr);
149         }
150     }
```

```
137         curr=head;
138         while(curr->data!=key && curr->link!=NULL)
139         {
140             prev=curr;
141             curr=curr->link;
142         }
143         if(curr->data!=key)
144             printf("Search data not found. Deletion is not possi
145         else
146         {
147             prev->link=curr->link;
148             free(curr);
149         }
150     }
151     Display();
152 }
153
```