```
1
 2
   // INSERTION AND DELETION FROM SINGLY LINKED LIST
3
   //calculate nth node from end in 2 pass
5 // Reversing the Linked List
 6 #include<stdio.h>
7
   #include<stdlib.h>
8
9 struct node
10 {
11
       int data;
12
       struct node *next;
13 };
14
15
   struct node *head=NULL;
16 int V,C;
17
18 void InsertInB(int Value) // insert the node in the Beginning
19
20
       V=Value;
21
       struct node *p=(struct node *)malloc(sizeof(struct node));
22
       p->data=V;
23
24
       if(head==NULL)
25
26
           head=p;
27
           p->next=NULL;
       }
28
29
       else
30
31
           struct node *q=head;
32
           p->next=q;
33
           head=p;
34
35
36
   }
37
   void InsertInE(int Value) // insert the node in the End
38
39
40
       V=Value;
41
       struct node *p=(struct node *)malloc(sizeof(struct node));
42
       p->data=V;
43
44
       if(head==NULL)
45
46
           head=p;
47
           p->next=NULL;
48
49
       struct node *q=head;
50
51
       while(q->next!=NULL)
52
53
          q=q->next;
54
55
       q->next=p;
56
       p->next=NULL;
57
58
59
   int LengthOfLL() // traverse LL to Calculate the length of LL
60
61
       C=0;
62
       struct node *p=head;
       while(p!=NULL)
63
64
65
               p=p->next;
66
               C++;
```

```
67
 68
         return C;
 69
 70
 71
    void InsertInM(int Value,int index) // insert the node in the middle of LL at
perticular Index.
 72 {
 73
         V=Value;
 74
         struct node *p=(struct node *)malloc(sizeof(struct node));
 75
         p->data=V;
 76
 77
         struct node *q=NULL, *r=NULL;
 78
         if(index>LengthOfLL())
 79
 80
             printf("incorrect Index Value");
         }
 81
 82
 83
         else
 84
 85
             int i;
 86
             q=head;
 87
             for(i=1;i<=index-1;i++)</pre>
 88
 89
                 r=q;
 90
                  q=q->next;
 91
 92
 93
             p->next=q;
 94
             r->next=p;
         }
 95
 96
 97
    }
 98
 99
    void DeleteFrmB() // delete the First Node in LL.
100
101
         struct node *p=head;
102
         head=head->next;
103
         free(p);
104
105
     void DeleteFrmE() // delete the Last Node in LL.
106
107
108
         struct node *p=head;
109
         while(p->next->next!=NULL)
110
111
             p=p->next;
112
113
114
         free(p->next);
115
         p->next=NULL;
116
117
118
    void DeleteFrmM(int index) // Delete the node from the middle at given index.
119
120
         int i;
121
         struct node *p=NULL,*q=NULL;
122
         p=head;
123
         for(i=1;i<=index-1;i++)</pre>
124
125
             q=p;
126
             p=p->next;
127
128
129
         q->next=p->next;
130
         free(p);
131
```

```
132
133
    void CalculateNthFromE(int index) // it will take two pass of scan to calculate...
134
135
        if(index>LengthOfLL()+1)
136
            printf("out of bound Index");
137
        else
138
139
            int i;
140
            struct node *p=head;
141
            int N=LengthOfLL()-index;
142
            for (i=1;i<=N;i++)</pre>
143
144
                p=p->next;
145
            printf("%d th node from last is %d\n",index,p->data);
146
         }
147
148
149
150
    void CalculateNthFromEin1Pass(int index)// Calculate Nth node from last in single
151
152
        struct node *p=head, *q=head;
153
154
        for (i=1;i<index;i++)</pre>
155
156
            p=p->next;
157
158
159
        while (p->next!=NULL)
160
161
            p=p->next;
162
            q=q->next;
163
164
        printf("\n\n%dth node from end in single pass %d\n",index,q->data);
165
166
    void reverseLL() // Reverse the Linked
167
168
        struct node *p=head, *q=NULL, *r=head;
169
170
171
            while(r!=NULL)
172
173
               r=r->next;
174
               p->next=q;
175
               q=p;
176
               p=r;
177
178
179
        head=q;
180
181
182
183
    184
    void PrintListFromEnd(struct node *head)
185
186
        if(!head)
187
            return;
188
        PrintListFromEnd(head->next);
        printf("%d",head->data);
189
190
        printf("--->");
191
    }
    /////// check given Linked List is Even or Odd in
193
    int CheckEvenOdd()
194
    {
```

```
195
         struct node *p=head;
196
         while(p&&p->next)
197
198
             p=p->next->next;
199
200
201
202
         if(p==NULL)
203
             printf("\n Even Length");
204
         else
205
             printf("\n\n Odd length");
206
207
208
209
210
211
212
    void displayLL() // display the complete LL created.
213
214
         struct node *dp=head;
215
         while(dp!=NULL)
216
217
             printf("%d",dp->data);
218
             printf("--->");
219
             dp=dp->next;
220
221
222
223
224
    void main()
225
226
         InsertInB(20);
227
         displayLL();
228
         printf("\n\n");
229
         InsertInB(30);
230
         InsertInB(40);
231
         displayLL();
         printf("\n\n");
232
233
         InsertInE(10);
234
         displayLL();
235
         printf("\n\n");
236
         InsertInM(500,3);displayLL();
237
         printf("\n\n");
238
         printf("Length of Current LL is %d\n", LengthOfLL());
239
240
        CalculateNthFromE(5);
241
        CalculateNthFromEin1Pass(4);
242
243
         DeleteFrmM(3);
244
         displayLL();
245
         printf("\n\n");
246
         DeleteFrmB();
247
         displayLL();
248
         printf("\n\n");
249
         DeleteFrmE();
250
         displayLL();
251
         printf("\n\n");
252
         InsertInB(12);InsertInB(32);InsertInB(21);InsertInB(65);
253
         displayLL();
254
         reverseLL();
255
         printf("\n\nReverseing the linked List\n\n");
256
         displayLL();
         printf("\n\n");
257
258
259
         CheckEvenOdd();
260
```