

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  struct node {
4      int data;
5      struct node *next;
6  };
7
8  struct node *start = NULL;
9  void insert_at_begin(int);
10 void insert_at_end(int);
11 void traverse();
12 void delete_from_begin();
13 void delete_from_end();
14 int count = 0;
15
16 int main () {
17     int i, data;
18
19     for (;;) {
20         printf("1. Insert an element at the beginning of linked list.\n");
21         printf("2. Insert an element at the end of linked list.\n");
22         printf("3. Traverse linked list.\n");
23         printf("4. Delete an element from beginning.\n");
24         printf("5. Delete an element from end.\n");
25         printf("6. Exit\n");
26
27         scanf("%d", &i);
28
29         if (i == 1) {
30             printf("Enter value of element\n");
31             scanf("%d", &data);
32             insert_at_begin(data);
33         }
34         else if (i == 2) {
35             printf("Enter value of element\n");
36             scanf("%d", &data);
37             insert_at_end(data);
38         }
39         else if (i == 3)
40             traverse();
41         else if (i == 4)
42             delete_from_begin();
43         else if (i == 5)
44             delete_from_end();
45         else if (i == 6)
46             break;
47         else
48             printf("Please enter valid input.\n");
49     }
50
51     return 0;
52 }
53
54 void insert_at_begin(int x) {
55     struct node *t;
56
57     t = (struct node*)malloc(sizeof(struct node));
58     t->data = x;
59     count++;
60
61     if (start == NULL) {
62         start = t;
63         start->next = NULL;
64     }
65     return;
66 }

```

```

67     t->next = start;
68     start = t;
69 }
70
71 void insert_at_end(int x) {
72     struct node *t, *temp;
73
74     t = (struct node*)malloc(sizeof(struct node));
75     t->data = x;
76     count++;
77
78     if (start == NULL) {
79         start = t;
80         start->next = NULL;
81         return;
82     }
83
84     temp = start;
85
86     while (temp->next != NULL)
87         temp = temp->next;
88
89     temp->next = t;
90     t->next = NULL;
91 }
92
93 void traverse() {
94     struct node *t;
95
96     t = start;
97
98     if (t == NULL) {
99         printf("Linked list is empty.\n");
100        return;
101    }
102
103    printf("There are %d elements in linked list.\n", count);
104
105    while (t->next != NULL) {
106        printf("%d\n", t->data);
107        t = t->next;
108    }
109    printf("%d\n", t->data); // Print last node
110 }
111
112 void delete_from_begin() {
113     struct node *t;
114     int n;
115
116     if (start == NULL) {
117         printf("Linked list is empty.\n");
118         return;
119     }
120
121     n = start->data;
122     t = start->next;
123     free(start);
124     start = t;
125     count--;
126
127     printf("%d deleted from the beginning successfully.\n", n);
128 }
129
130 void delete_from_end() {
131     struct node *t, *u;
132     int n;

```

```
133
134     if (start == NULL) {
135         printf("Linked list is empty.\n");
136         return;
137     }
138
139     count--;
140
141     if (start->next == NULL) {
142         n = start->data;
143         free(start);
144         start = NULL;
145         printf("%d deleted from end successfully.\n", n);
146         return;
147     }
148
149     t = start;
150
151     while (t->next != NULL) {
152         u = t;
153         t = t->next;
154     }
155
156     n = t->data;
157     u->next = NULL;
158     free(t);
159
160     printf("%d deleted from end successfully.\n", n);
161 }
```