

```
1 #include <inttypes.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 struct Node {
6     int data;
7     struct Node* nxp;
8 };
9
10 struct Node* XOR(struct Node* a, struct Node* b)
11 {
12     return (struct Node*)((uintptr_t)(a) ^ (uintptr_t)(b));
13 }
14
15 struct Node* insert(struct Node** head, int value)
16 {
17     if (*head == NULL) {
18         struct Node* node
19             = (struct Node*)malloc(
20                 sizeof(struct Node));
21
22         node->data = value;
23         node->nxp = XOR(NULL, NULL);
24         *head = node;
25     }
26     else {
27         struct Node* curr = *head;
28         struct Node* prev = NULL;
29         struct Node* node = (struct Node*)malloc(sizeof(struct Node));
30         curr->nxp = XOR(node, XOR(NULL, curr->nxp));
31         node->nxp = XOR(NULL, curr);
32         *head = node;
33         node->data = value;
34     }
35     return *head;
36 }
37
38 void printList(struct Node** head)
39 {
40     struct Node* curr = *head;
41     struct Node* prev = NULL;
42     struct Node* next;
43     while (curr != NULL) {
44         printf("%d ", curr->data);
45         next = XOR(prev, curr->nxp);
46         prev = curr;
47         curr = next;
48     }
49 }
50
51 struct Node* delEnd(struct Node** head)
52 {
53     if (*head == NULL)
54         printf("List is empty\n");
55     else {
56         struct Node* curr = *head;
57         struct Node* prev = NULL;
58         struct Node* next;
59         while (XOR(curr->nxp, prev) != NULL) {
60             next = XOR(prev, curr->nxp);
61             prev = curr;
62             curr = next;
63         }
64         if (prev != NULL)
65             prev->nxp = XOR(XOR(prev->nxp, curr), NULL);
66         else
67             *head = NULL;
68         free(curr);
69         printf("Deleted Last Element!\n");
70     }
71     return *head;
72 }
73
74 int main ()
75 {
76     printf("XOR Linked List\n");
77     int op,a;
78     struct Node *head = NULL;
79     for(;;){
80         printf("\n1.Insert\n2.Delete\n3.Display\n4.Exit\nEnter your choice:");
81         scanf( "%d",&op);
82
83         switch(op){
84             case 1:
85                 printf("Enter Element:");
86                 scanf("%d",&a);
87                 insert(&head, a);
88                 printf("Element Inserted!\n");
89                 break;
90             case 2:
91                 delEnd(&head);
92                 break;
93             case 3:
94                 if (head == NULL)
95                     printf("List is empty\n");
96                 else{
97                     printList(&head);
98                     printf("\n");
99                 }
100                 break;
101             case 4:
102                 exit(0);
103             default:
104                 printf("Invalid Choice!");
105                 break;
106         }
107     }
108 }
```

OR Linked List

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice:1

Enter Element:50

Element Inserted!

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice:1

Enter Element:60

Element Inserted!

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice:1

Enter Element:70

Element Inserted!

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice:3

70 60 50

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice:2

Deleted Last Element!

1.Insert

2.Delete

3.Display

4.Exit

Enter your choice:2