

```

#include <bits/stdc++.h>
using namespace std;

1  class Node {
2      public:
3          int data;
4          Node *next;
5  };
6
7  class LinkedList{
8      private:
9          Node *head;
10         // the above is a pointer var of the type Node and is confined to a link obj..
11     public:
12         void add_node(int data);
13         int search(int data);
14         int length();
15         void display();
16         int Delete(int data);
17         void insert(int data, int pos);
18         void push_node(int data);
19         bool isempty();
20
21 };
22
23 void LinkedList::add_node(int data){
24
25     Node *new_node = new Node(); // cretaing a node obj
26     new_node->data = data;
27     new_node->next = NULL;
28     // if we are adding a first node then make the head points to it...
29     // else iterate the list unutil reach the end and set the last node pointer to the current;
30     if(head == NULL){
31         head = new_node; // setting the head pointer to new_node.
32     }
33     else{
34         Node *temp = head;
35         while(temp->next!=NULL){
36             temp = temp->next ;
37         }
38         temp->next = new_node; // setting the last node to the newnode
39     }
40 }
41
42 void LinkedList::display(){
43     if (head == NULL){
44         cout << "List is Empty !" << endl;
45     }
46     else{
47         cout << "List is not empty !\n";
48         Node *begin = new Node();
49         begin = head; // point the begin pointer of type node to the head pointer..
50         while(begin != NULL){
51             cout << "| " << begin->data<< " | * ----> ";
52             begin = begin->next;
53         }
54         cout << "NULL";
55         cout << endl;
56     }
57 }
58
59 int LinkedList::search(int data){
60     Node *begin = new Node();
61     begin = head;
62     int pos = 0;
63
64     while(begin!= NULL){
65         pos ++;
66         if(begin->data == data){
67

```

```

67         return pos;
68     }
69     else{
70         begin = begin->next;
71     }
72 }
73 return -1;
74 }
75
76 int LinkedList::length(){
77     Node *begin = new Node();
78     begin = head;
79     int pos = 0;
80     while(begin!=NULL){
81         begin = begin->next;
82         pos ++;
83     }
84     return pos;
85 }
86
87 void LinkedList::push_node (int data){
88     Node *new_node = NULL;
89     new_node = new Node();
90     new_node->data = data;
91     // the new_node points to the node which was earlier pointed by head...
92     // ie. address was stores in Head node var
93     new_node->next = head;
94     // head points to the new_node now it is like  head----> new_node----> address in head earlier
95     head = new_node;
96 }
97
98 void LinkedList::insert(int data, int index){
99     Node *new_node = new Node();
100     Node *present = new Node();
101     present = head;
102     new_node->data = data;
103     if(index == 0) push_node(data);
104     else{
105
106         for (int i = 1; present->next !=NULL && i< index-1 ; i++) {
107             present = present->next;
108         }
109
110         new_node->next= present->next;
111         present->next = new_node;
112     }
113 }
114
115 int LinkedList::Delete(int k){
116     int index =1;
117     int value_deleted = 0;
118     // crreated two poniter ton nodes , present and next
119     Node *present = new Node();
120     Node *next = new Node();
121     // set present to the head
122     present = head;
123     // if head is to be removed set the head to the next element
124     if(k ==0){
125         head = present->next;
126         // use free keyword to free the space
127         free(present);
128         return present->data;
129     }
130
131     // if any other pos is to be removed iterate until that pos -1 and set present to point to it.
132     else {
133         for(int i =1; present->next!=NULL && i< k-1 ; i++){
134             present = present->next;
135         }
136     }
137     value_deleted = present->next->data;

```

```

138     next = present->next->next;
139     free(present->next);
140     present->next = next;
141     return value_deleted;
142 }
143
144 bool LinkedList::isempty(){
145     if(head == NULL)
146         return true;
147     else
148         return false;
149 }
150
151 int main(){
152
153     LinkedList * list = new LinkedList();
154     int ch=0;
155     int index =0;
156     int value=0;
157     int n =0;
158
159     cout << "1. Enter the elements in the list \n";
160     cout << "2. Search for an element in the list \n";
161     cout << "3. Enter an element at the kth index of created list \n";
162     cout << "4. Delete an element at the kth index of created list \n";
163     cout << "5. To find the length of the string \n";
164     cout << "6. Display list \n";
165     cout << "7 Is list empty \n";
166
167
168     while(true){
169         cout << "\n Enter your choice : ";
170         cin >> ch;
171
172         switch(ch){
173
174             case 1:
175                 // Add simple
176                 cout << "Enter the no of elements you want to enter into the LinkedList randomly (INSERTION IN PROCEEDINGS BELOW): ";
177                 cin >> n;
178                 for(int i =0; i < n;i++){
179                     cout << "Enter the value to be added : ";
180                     cin >> value;
181                     list->add_node(value);
182                 }
183                 break;
184             case 2:
185                 //Search
186                 cout << "Enter a value to search in the linked list: ";
187                 cin >> value;
188
189                 if (list->search(value) != -1)
190                     cout << "Found at pos " << list->search(value) << "\n";
191                 else
192                     cout << "Not found in the list \n";
193                 break;
194             case 3:
195
196                 // Adding element at kth pos
197                 cout << "Enter the index at which element is to be added:";
198                 cin >> index;
199                 cout << "Enter the value to be added .:";
200                 cin >> value;
201
202                 if (index ==1 || 0){
203                     list->push_node(value);
204                 }
205                 else{

```

```

208     list->insert(value,index);
209 }
210 cout << "\n";
211 list->display();
212
213 break;
214 case 4:
215     // Delete
216     cout << "Enter the index of element to be deleted : ";
217     cin >> index;
218     cout << "Deleted value is : " << list->Delete(index) << "\n ";
219     list->display();
220     break;
221 case 5:
222     // Length of list
223     cout << "\n";
224     cout << "Length of the link list is : " << list->length() << "\n ";
225     break;
226
227 case 6:
228     // Display
229     list->display();
230     break;
231
232 case 7:
233     bool value;
234     cout << "Cheking List is empty : ";
235     value = list->isempty();
236     if ( value == false){
237         cout << "false";
238     }
239     else cout << "true";
240     cout << "\n";
241     break;
242 default:
243     list->display();
244     cout << "Enter a valid choice ";
245 }
246
247
248
249 }
250

```