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

class Node {
    public:
        int data;
        Node *next;
};

class LinkedList{
    private:
        Node *head;
        // the above is a pointer var of the  type Node and is confined to a link obj..
    public:
        void add_node(int data);
        int search(int data);
        int length();
        void display();
        int Delete(int data);
        void insert(int data, int pos);
        void push_node(int data);
        bool isempty();

};

void LinkedList::add_node(int data){

    Node *new_node = new Node();  // cretaing a node obj
    new_node->data = data;
    new_node->next = NULL;
    //  if we are adding a first node then make the head points to it...
    //  else iterate the list unutil reach the end and set the last node pointer to the current;
    if(head == NULL){
        head = new_node; // setting the head pointer to new_node.
    }
    else{
        Node *temp = head;
        while(temp->next!=NULL){
            temp = temp->next ;
        }
        temp->next = new_node; // setting the last node to the newnode
    }
}

void LinkedList::display(){
    if (head == NULL){
        cout << "List is Empty !" << endl;
    }
    else{
    cout << "List is not empty ! \n";
    Node *begin = new Node();
    begin = head; // point the begin pointer of type node to the head pointer..
    while(begin != NULL){
         cout << "| " << begin->data<< " | * ----> ";
        begin = begin->next;
        }
        cout << "NULL";
        cout << endl;
    }
}

int LinkedList::search(int data){
    Node *begin = new Node();
    begin = head;
    int pos = 0;
    while(begin!= NULL){
        pos ++;
        if(begin->data == data){
```

```
            return pos;
        }
        else{
            begin = begin->next;
        }
    }
    return -1;
}

int LinkedList::length(){
    Node *begin = new Node();
    begin = head;
    int pos = 0;
    while(begin!=NULL){
        begin = begin->next;
        pos ++;
    }
   return pos;
}

void LinkedList::push_node (int data){
    Node *new_node = NULL;
    new_node = new Node();
    new_node->data = data;
    // the new _node points to the node which was earlier pointed by head...
    // ie. addess was stores in Head node var
    new_node->next = head;
    // head points to the new_node now it is like   head---> new_node----> address in head earlier
    head = new_node;
}

void LinkedList::insert(int data, int index){
    Node *new_node = new Node();
    Node *present = new Node();
    present = head;
    new_node->data = data;
 if(index == 0) push_node(data);
 else{

     for (int i = 1; present->next !=NULL &&  i< index-1 ; i++) {
         present = present->next;
     }

        new_node->next= present->next;
        present->next = new_node;
 }
}

int LinkedList::Delete(int k){
    int index =1;
    int value_deleted = 0;
    // crreated two poniter ton nodes , present and next
    Node *present = new Node();
    Node *next = new Node();
    // set present to the head
    present = head;
    // if head is to be removed set the head to the next element
    if(k ==0){
        head = present->next;
        // use free keyword to free the space
        free(present);
        return present->data;
    }
    // if any other pos is to be removed  iterate until that pos -1 and set present to point to it.
    else {
        for(int i =1; present->next!=NULL &&  i<k-1 ; i++){
            present = present->next;
        }
    }
    value_deleted = present->next->data;
```

```
    next = present->next->next;
    free(present->next);
    present->next = next;
    return value_deleted;
}

bool LinkedList::isempty(){
    if(head == NULL)
    return true;
    else
    return false;
}

int main(){

  LinkedList * list = new LinkedList();
    int ch=0;
    int index =0;
    int value=0;
    int n =0;

    cout << "1. Enter the elements in the list \n";
    cout << "2. Search for an element in the list \n";
    cout << "3. Enter an element at the kth index of created list \n";
    cout << "4. Delete an element at the kth index of created list \n";
    cout << "5. To find the length of the string \n";
    cout << "6. Display list \n";
    cout << "7 Is list empty \n";


while(true){
    cout << " \n Enter your choice : ";
    cin >> ch;

    switch(ch){

    case 1:
    // Add simple
    cout << "Enter the no of elements you want to enter into the LinkedList randomly (INSERTION IN PROCEEDINGS BELOW): ";
    cin >>n;
    for(int i =0; i < n;i++){
        cout << "Enter the value to be added : ";
        cin >> value;
        list->add_node(value);

    }
    break;
    case 2:
    //Search
        cout << "Enter a value to search in the linked list: ";
        cin  >> value;

        if (list->search(value) != -1)
            cout << "Found at pos " << list->search(value) << "\n";
        else
            cout << "Not found in the list \n";
    break;
    case 3:

      // Adding element at kth pos
    cout << "Enter the index at which element is to be added:";
    cin >> index;
    cout << "Enter the value to be added :";
    cin >> value;

    if (index ==1 || 0){
        list->push_node(value);
    }
    else{
```

```
        list->insert(value,index);
    }
    cout << "\n";
    list->display();

    break;
    case 4:
    // Delete
    cout << "Enter the index of element to be deleted : ";
    cin  >> index;
    cout << "Deleted value is : " <<  list->Delete(index) << " \n ";
    list->display();
    break;
    case 5:
    // Length of list
    cout << "\n";
    cout << "Length of the link list is : " << list->length() << "\n ";
    break;

    case 6:
    // Display
      list->display();
        break;

    case 7:
        bool value;
        cout << "Cheking List is empty : ";
        value = list->isempty();
        if ( value == false){
            cout << "false";
        }
        else cout << "true";
        cout << "\n";
        break;
    default:
        list->display();
        cout << "Enter a valid choice ";
}


 }


}
```