// Dealing with a singly-linked list
// Written by:
// Date: August 27, 2008

#include <iostream>
using namespace std;

// Following is the blueprint of a node on the linked list
struct nodeType
{
    int info;
    nodeType* link;
};

// Function prototypes
nodeType* buildList();
// This function reads a sequence of integers ending with -999,
// builds a linked list of nodes holding those integers in order,
// and returns the pointer to the first node of the list.

void printList(nodeType*);
// This function takes the head pointer to a list, and systematically
// prints the integers appearing on the list.

int length(nodeType*);
// This function takes the head pointer to a list, and returns
// the length of the list, i.e., number of elements on the list

bool isPresent(nodeType* , int);
// This function determines whether a particular integer appears
// on the list.
// In: Head pointer of the list and the integer being searched
// Out: true if the given ineteger appears on the list and
//      false, otherwise.

int main()
{
    int item; // input integer
    nodeType* head; // Pointer to the first node of the list

    // First build the list
    head = buildList();

    // Next, print the integers appearing on the list
cout << "Elements on the list are: ";
printList(head);
cout << endl;

    // Display the length of the list
cout << "Length of the list is: " << length(head)
    << endl << endl;

    // Do a search for an element on the list
cout << "Please enter an item to be searched: ";
cin >> item;
if (isPresent(head, item))
    cout << "The integer " << item
    << " is present on the list." << endl;
else
    cout << "The integer " << item
    << " is not present on the list. " << endl;
    cout << endl;
return 0;
} // end of main

nodeType* buildList()
{
    nodeType* first; // pointer to the first node of the list
    nodeType* last; // pointer to the last node of the list
    nodeType* p; // local pointer variable
    int num;

cout << "Please enter a list of integers ending with -999:" << endl;
    cin >> num;
    if(num == -999)
        return NULL; // in this case the list is empty
    else // build the first node
    {
        first = new nodeType;
        first->info = num;
        first->link = NULL;
        last = first;
    }

    // Add the subsequent nodes
    cin >> num;
    while (num != -999)
    {
        p = new nodeType;
        p->info = num;
        p->link = NULL;

        last->link = p;
        last = p;
        cin >> num;
    } // end while
    return first;
} // end of buildList

void printList(nodeType* head)
{
    nodeType* p;
    p = head;

    while (p != NULL)
    {
        cout << p->info << " ";
        p = p->link;
    } // end of while
} // end of printList
int length(nodeType* p)
{
    int count = 0;
    while (p != NULL)
    {
        count++;
        p = p->link;
    }
    return count;
} // end of length

bool isPresent(nodeType* head, int target)
{
    if (head == NULL) // in this case, the list is empty.
        return false;

    // In what follows, the list is non-empty.
    bool found = false;
    nodeType* p;
    p = head;

    while (p != NULL)
    {
        if (p->info == target)
        {
            found = true;
            break;
        }
        else
            p = p->link;
    } // end of while
    return found;
} // end of isPresent