```
//
//  Display 17.14 Interface File for a Linked List Library
//  main.cpp
//  AbsoluteCpp_ch17_14
//

//This is the header file listtools.h. This contains type definitions
//and function declarations for manipulating a linked list to store
//data of any type T. The linked list is given as a pointer of type
//Node<T>* that points to the head (first) node of the list. The
//implementation of the functions is given in the file listtools.cpp

#ifndef LISTTOOLS_H
#define LISTTOOLS_H

namespace LinkedListSavitch
{
template<class T>
class Node
{
public:
    Node(const T& theData, Node<T>* theLink) : data(theData),
    link(theLink){}
    Node<T>* getLink( ) const { return link; }
    const T getData( ) const { return data; }
    void setData(const T& theData) { data = theData; }
    void setLink(Node<T>* pointer) { link = pointer; }
private:
    T data;
    Node<T> *link;
};

template<class T>
void headInsert(Node<T>*& head, const T& theData);
//Precondition: The pointer variable head points to
//the head of a linked list.
//Postcondition: A new node containing theData
//has been added at the head of the linked list.

template<class T>
void insert(Node<T>* afterMe, const T& theData);
//Precondition: afterMe points to a node in a linked list.
//Postcondition: A new node containing theData
//has been added after the node pointed to by afterMe.

template<class T>
void deleteNode(Node<T>* before);
//Precondition: The pointer before points to a node that has
//at least one node after it in the linked list.
//Postcondition: The node after the node pointed to by before
//has been removed from the linked list and its storage
//returned to the freestore.

template<class T>
void deleteFirstNode(Node<T>*& head);
```

```
//Precondition: The pointer head points to the first
//node in a linked list with at least one node.
//Postcondition: The node pointed to by head has been removed
//from the linked list and its storage returned to the freestore.

template<class T>
Node<T>* search(Node<T>* head, const T& target);
//Precondition: The pointer head points to the head of a linked list.
//The pointer variable in the last node is NULL.
//== is defined for type T.
//(== is used as the criterion for being equal.)
//If the list is empty, then head is NULL.
//Returns a pointer that points to the first node that
//is equal to the target. If no node equals the target,
//then the function returns NULL.
}//LinkedListSavitch

#endif //LISTTOOLS_H
```

```
//
//  Display 17.15 Implementation File for a Linked List Library
//  listtools.cpp
//  AbsoluteCpp_ch17_14
//

//This is the implementation file listtools.cpp. This file contains
//function definitions for the functions declared in listtools.h.

#include <cstddef>
#include "listtools.h"

namespace LinkedListSavitch {

    template<class T>
    void headInsert(Node<T>*& head, const T& theData)
    {
        head = new Node<T>(theData, head);
    }

    template<class T>
    void insert(Node<T>* afterMe, const T& theData)
    {
        afterMe->setLink(new Node<T>(theData, afterMe->getLink( )));

    }

    template<class T>
    void deleteNode(Node<T>* before)
    {
        Node<T> *discard;
        discard = before->getLink( );
        before->setLink(discard->getLink( ));
        delete discard;
    }

    template<class T>
    void deleteFirstNode(Node<T>*& head)
    {
        Node<T> *discard;
        discard = head;
        head = head->getLink( );
        delete discard;
    }

    //Uses cstddef:
    template<class T>
    Node<T>* search(Node<T>* head, const T& target)
    {
        Node<T>* here = head;
        if (here == NULL) //if empty list
        {
            return NULL;
        }
        else
```

```
        {
            while (here->getData( ) != target && here->getLink( ) != NULL)
             here = here->getLink( );

            if (here->getData( ) == target) return here;
            else
                return NULL;
        }
    }
}//LinkedListSavitch
```