```
*********************************************************************************
*********************************************************************************
**                                                                             **
**                             deletetail.cc listing                           **
**                                                                             **
*********************************************************************************
*********************************************************************************

#include <iostream>
#undef NULL
const int NULL = 0;
using namespace std;
/*
        Steven Liu
        CS215−J001
        Spring, 2011
        Extra Credit − LList::DeleteTail()

        This program demonstrates deletion of "tail" listnode from a
        non−empty linked list.  User will enter a couple of element
        into the linkedlist, then the last two listnodes will
        be deleted using the DeleteTail() method.
*/


//*******************************global section*******************************
typedef int element;            //datatype of "element"
const element SENTINEL = −1;    //"element value" that ends user input


//reads single type checked element
element read_element();


//listnode class
        //each listnode consists of 2 sides:
        //1) one side, called "data" holds a single element
        //2) the other side, called "next" holds the address of the
        //next listnode
class listnode {
        public:
                element data;   //holds actual data
                listnode * next;        //holds address of next listnode
        };


//Linked List class
        //a valid linked list is defined as:
        //1) "head" points to the first listnode
        //2) followed by a series of listnodes
        //3) last listnode pointing to NULL
        //4) "tail" points to last listnode
        //when the list is empty (but also valid):
        //1) "head" points to NULL
        //2) "tail" is undefined
class LList {
        private:
                listnode * head;                //points to the first listnode
                listnode * tail;                //points to the last listnode
        public:
                //constructor/destructor:
                LList();        //constructor − auto called upon N.O. birth
                ~LList();       //destructor − auto called before N.O. death
                //methods:
                void Clean();
                void Print();
                void ReadForward();
                element DeleteTail();   //extra credit
        };
```

```

//−−−−−−−−−−−−−−−−−−−−−−−−−−−−End global section−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

//******************************MAIN FUNCTION*****************************




//**main function**
int main(){
        LList myLList;

        myLList.ReadForward();

        myLList.Print();

        myLList.DeleteTail();
        myLList.DeleteTail();

        myLList.Print();
        }



//−−−−−−−−−−−−−−−−−−−−−−−−−−−−END MAIN FUNCTION−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

//******************************global functions*****************************


//type checks input to see if it matches "element"
element read_element() {
        //variable dec+def
        element user_input;     //input − user input

        //type checking
        cin >> user_input;
        while (!cin.good()){
                cout << "Bad input datatype; Try again: ";
                cin.clear();
                cin.ignore(80, ’\n’);
                cin >> user_input;
                }

        return user_input;
        }


//−−−−−−−−−−−−−−−−−−−−−−−−−−−−End global functions−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

//************************LList constructor/destructor************************


//constructor
LList::LList(){
        //pre: none
        //post: the N.O. LList is empty
        head = NULL;
        }


//destructor
LList::~LList(){
        //pre: the N.O. LList is valid
        //post: the N.O. LList is empty
        Clean();
        }
```

```
//------------------End LList constructor/destructor-----------------------

//****************************LList methods*********************************


//cleans the LList of all nodes
void LList::Clean(){
        //pre: N.O. is valid
        //post: N.O. is now empty and all of its former listnodes have
                //had their memory returned to the system memory pool

        listnode * temp;                //points listnode to be deleted

        //we point "head" at the next listnode, maintaining a valid LList
        //while "temp" points to the listnode we want to delete
        while (head != NULL) {
                temp = head;
                head = head->next;
                delete temp;
                }
        }


//prints out the entire LList
void LList::Print(){
        //pre: N.O. is valid
        //post: N.O. is unchanged, and the element it contains
                //have been displayed

        //LCV - begins at head then traverses entire LList
        listnode * temp;

        temp = head;
        while (temp != NULL) {
                cout << temp->data << " ";
                temp = temp->next;      //pointer increment
                }
        cout << endl;
        }


//reads in data, and puts new data at the END of linked list
void LList::ReadForward(){
        //pre: N.O. is valid
        //post: N.O. is valid, containing elements entered by user
                //in forward order

        Clean();        //removes any existing listnodes in linked list

        element userval;        //input/LCV - stores user element input
        listnode * temp;                //keeps track of new listnode

        cout << "Enter elements, " << SENTINEL << " to stop: ";
        userval = read_element();
        while (userval != SENTINEL){
                temp = new listnode;
                temp->data = userval;
                temp->next = NULL;
                if (head == NULL)       //first time
                        head = temp;
                else //not first time
                        tail->next = temp;
                tail = temp;
                userval = read_element();
                }
        }
```

```
//removes last listnode in the list and returns the element in the listnode
element LList::DeleteTail() {   //extra credit
        //pre: N.O. is valid and non-empty
        //post: N.O. is unchanged, except the listnode at the tail-end
                //has been removed, its memory returned to the system pool,
                //called heap, and its element returned to the caller

        element val;            //holds element in listnode to be deleted
        listnode * temp;                //points to the listnode to be deleted

        temp = head;
        while (temp->next != tail)
                temp = temp->next;
        //by end of loop we know:
                //"temp" is now pointing to the second to last listnode

        val = tail->data;
        delete tail;
        tail = temp;
        tail->next = NULL;
        return val;
        }
```

```
*******************************************************************************
*******************************************************************************
**                                                                           **
**                          deletetail.cc compilation                        **
**                                                                           **
*******************************************************************************
*******************************************************************************

c++ compilation succeeded
```

```
*******************************************************************************
*******************************************************************************
**                                                                           **
**            deletetail.cc execution − unstructured testcase 1 [#1]         **
**                                                                           **
*******************************************************************************
*******************************************************************************

Enter elements, −1 to stop: 1 2 3 4 56 7 −1
1 2 3 4 56 7
1 2 3 4
```

```
*******************************************************************************
*******************************************************************************
**                                                                           **
**           deletetail.cc execution − unstructured testcase 0 [#2]          **
**                                                                           **
*******************************************************************************
*******************************************************************************

Enter elements, −1 to stop: 1
2
3
4
5
6
−1
1 2 3 4 5 6
1 2 3 4
```

```
*******************************************************************************
*******************************************************************************
**                                                                           **
**           deletetail.cc execution − unstructured testcase 2 [#3]          **
**                                                                           **
*******************************************************************************
*******************************************************************************

Enter elements, −1 to stop: 9
8
7
6 5 4 3 2 1 −1
9 8 7 6 5 4 3 2 1
9 8 7 6 5 4 3
```

```
********************************************************************************
********************************************************************************
**                                                                            **
**           deletetail.cc execution − unstructured testcase 4 [#4]           **
**                                                                            **
********************************************************************************
********************************************************************************

Enter elements, −1 to stop: abc
Bad input datatype; Try again: 123 123 123 123
−1
123 123 123 123
123 123
```