```
******************************************************************************
******************************************************************************
**                                                                          **
**                         reverseinplace.cc listing                        **
**                                                                          **
******************************************************************************
******************************************************************************

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


//******************************global section******************************
typedef int element;            //datatype of "element"
const element SENTINEL = −1;    //value of element 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 to the
        //next listnode
class listnode {
        public:
                element data;           //holds actual data
                listnode * next;        //holds address to 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();
                void ReserveInPlace();  //extra credit
        };


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

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

```




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

        myLList.ReadForward();

        myLList.Print();

        myLList.ReserveInPlace();

        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();
                }
        }


//reverses the listnodes in the N.O. LList - cannot use extra memory space
void LList::ReserveInPlace() {  //extra credit
        //pre: the N.O. is valid
        //post: the N.O. is unchanged, except elements in its listnodes
```

```
                //are now in reverse order

        if ((head != NULL) && (head->next != NULL)) {
                //since we're inside of the if statement,
                        //there MUST be at least 2 listnodes in the LList

                listnode * prev;                //points to previous listnode
                listnode * curr;                //points to current listnode
                listnode * succ;                //points to succeeding listnode

                prev = head;
                curr = head->next;
                succ = curr->next;

                //since there are at least 2 listnodes, we have to reverse
                        //listnodes (loop body) at least once - dowhile loop
                //we're done when:
                        //prev == tail OR curr == NULL, only need to pick one
                        //becase we increment both prev and curr every loop
                do {
                        curr->next = prev;      //reverse listnode

                        //pointer increments:
                        prev = curr;
                        curr = succ;
                        if (succ != NULL)
                                succ = succ->next;
                        else
                                ;
                        } while (prev != tail);
                //by end of the above loop we know:
                        //directions of all listnodes have been reversed
                        //but the two ends of the listnodes aren't clear
                //however, we know that:
                        //1) head is currently pointing to new tail
                        //2) tail is currently pointing to new head
                        //3) prev is also pointing to new head

                tail = head;
                tail->next = NULL;
                head = prev;
                }
        else
                cout << endl << "LList has less than 2 listnodes "
                        << "and is therefore already ordered." << endl;
        }


//----------------------------End LList methods------------------------------
```

```
********************************************************************************
********************************************************************************
**                                                                            **
**                       reverseinplace.cc compilation                        **
**                                                                            **
********************************************************************************
********************************************************************************

c++ compilation succeeded
```

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

Enter elements, −1 to stop: a
Bad input datatype; Try again: b
Bad input datatype; Try again: −1


LList has less than 2 listnodes and is therefore already ordered.
```

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

Enter elements, −1 to stop: 1 −1
1

LList has less than 2 listnodes and is therefore already ordered.
1
```

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

Enter elements, −1 to stop: −1


LList has less than 2 listnodes and is therefore already ordered.
```

```
********************************************************************************
********************************************************************************
**                                                                            **
**      reverseinplace.cc execution − unstructured testcase 1 2 −1 [#4]       **
**                                                                            **
********************************************************************************
********************************************************************************

Enter elements, −1 to stop: 1 2 −1
1 2
2 1
```

```
********************************************************************************
********************************************************************************
**                                                                            **
**    reverseinplace.cc execution − unstructured testcase more input [#5]     **
**                                                                            **
********************************************************************************
********************************************************************************

Enter elements, −1 to stop: 1 2 3 4 5 6 7 8 9 0 −1
1 2 3 4 5 6 7 8 9 0
0 9 8 7 6 5 4 3 2 1
```

```
******************************************************************************
******************************************************************************
**                                                                          **
**   reverseinplace.cc execution − unstructured testcase random inputs [#6]   **
**                                                                          **
******************************************************************************
******************************************************************************

Enter elements, −1 to stop: 64  65 5 4 6 0 3 564 8 4  3 45 −1
64 65 5 4 6 0 3 564 8 4 3 45
45 3 4 8 564 3 0 6 4 5 65 64
```