```
********************************************************************************
********************************************************************************
**                                                                            **
**                              alist.cc listing                              **
**                                                                            **
********************************************************************************
********************************************************************************

#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;

/*
        Steven Liu
        CS215-J001
        Spring, 2011
        Program 1

1. Program description:
        This program will display and prompt the user to choose from eight
        different options.  Two of these options will allow the user to
        generate a list, three options will sort the list, two options will
        search through the list, and the final option will exit the program.
2. Precondition:
        i.      User will enter the main menu option, called menu_option, which
                will be a whole number between 0 and 8.
        ii.     When the user chooses to reset the list using user input, the
                user will enter each element into the element called
                element_input.
        iii.    When the user chooses to reset the list using randomly
                generated elements, the user will enter the size of the list,
                a whole number called list_size.  The user will also limit
                the range of the possible randomly generated elements by giving
                the lower limit, a whole number called range_low, and the upper
                limit, a whole number called range_high.
        iv.     When the user chooses to search the list, the user will provide
                the target to search far, a valid element called search_target.
3. Postcondition:
        i.      If the user chooses to reset the list, the program will
                create a new list from either user input or by randomly
                generating the elements in the list, then will display the
                contents of the list.  The program will also display a
                warning that the newly reset list is not known to be ordered.
        ii.     If the user chooses to sort the list, the program will first
                sort the list, then display the theoretical number of
                comparisons and movements made by the sort, then the actual
                number of comparisons and movements made by the program during
                the sort.  The program will then display contents of the list
                and will report that the list is now known to be ordered.
        iii.    If the user chooses to search through the list,
                the program will display whether the search target was
                found/not found.  The program will only attempt binary search
                if the list is known to be ordered.  If the target was on the
                list, the program will report the position of the first
                location the target was found.  Finally the program will
                display the number of comparisons made during the search.
        iv.     If the user chooses to quit, the program will exit.
*/


//global section:
const int MLS = 50;             //size of array
typedef int element;            //datatype of "element"
const element SENTINEL = -1;    //"element value" that ends user input



//global function prototypes:
```



```
element read_element();
int read_int();
void display_menu();
//array list class prototype:
class AList {
        private:
                element items[MLS];
                int size;
                bool list_ordered;
                //-I/O:
                void Print();
                void Read();
                void GenRandList(int list_size, int range_high, int range_low);
                void Swap(element first, element second);
                //-Sorts:
                void BubbleSort(int & comps, int & moves);
                void InsertionSort(int & comps, int & moves);
                void SelectionSort(int & comps, int & moves);
                //-Searches:
                void LinearSearch(element target, bool & found, int & position,
                        int & comps);
                void BinarySearch(element target, bool & found, int & position,
                        int & comps);
                //-Stats:
                int CalcQuad();
                int CalcLog();
        public:
                void FirstLoad();
                void Run_Print();
                void Run_Read();
                void Run_GenRandList();
                void Run_BubbleSort();
                void Run_InsertionSort();
                void Run_SelectionSort();
                void Run_LinearSearch();
                void Run_BinarySearch();
        };


//****main function****
int main() {
        cout << "Sort and Search Demo Program, version 1.0" << endl
                << "(c) 2011, (Steven Liu)" << endl << endl;

        srand(int(time(0)));    //seed the random number generator

        AList myAList;          //create object
        myAList.FirstLoad();    //prepares object for use
        int menu_option;        //input - user menu choice
        //loop menu
        do {
                myAList.Run_Print();
                display_menu();
                menu_option = read_int();

                switch (menu_option) {
                        case 1:
                                myAList.Run_Read();
                                break;
                        case 2:
                                myAList.Run_GenRandList();
                                break;
                        case 3:
                                myAList.Run_BubbleSort();
                                break;
                        case 4:
                                myAList.Run_InsertionSort();
                                break;
```

```
                        case 5:
                                myAList.Run_SelectionSort();
                                break;
                        case 6:
                                myAList.Run_LinearSearch();
                                break;
                        case 7:
                                myAList.Run_BinarySearch();
                                break;
                        case 8:
                                cout << endl << "Quitting Sort and Search "
                                        << "Demo Program, version 1.0" << endl;
                                break;
                        default:
                                cout << endl << "Invalid choice; Please choose"
                                        << "between menu options 1-8" << endl
                                        << endl;
                                break;
                        }
                } while (menu_option != 8);
        }




//-----------------------------Global Functions-----------------------------



//type checks input to see if it matches "element"
//if element is int, will also make certain
//range is between -2147483648 and 2147483648
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;
        }




//type checks input to ensure it is an integer
int read_int() {
        //variable dec+def
        int user_input;         //input - user input

        //type checking
        cin >> user_input;
        while (!cin.good()){
                cout << "Response must be a whole number, try again: ";
                cin.clear();
                cin.ignore(80, '\n');
                cin >> user_input;
                }

        return user_input;
```



```
        }




//displays the main menu to the user
void display_menu() {
        cout << "Actions:" << endl
                << "   1.  Reset the current list from the keyboard" << endl
                << "   2.  Reset the current list using "
                << "randomly generated elements" << endl
                << "   3.  Perform Bubble Sort on the current list" << endl
                << "   4.  Perform Insertion Sort on the current list" << endl
                << "   5.  Perform Selection Sort on the current list" << endl
                << "   6.  Perform Linear Search on the current list" << endl
                << "   7.  Perform Binary Search on the current list" << endl
                << "   8.  Quit the program" << endl << endl
                << "Choose an action: ";
        }




//*****************************End Global Functions************************

//-----------------------------I/O Methods----------------------------------



//prints the entire contents of the list
void AList::Print() {
        //Pre: the Native Object AList is valid
        //Post: the Native Object AList is unchanged, and its elements are
                //displayed
        for (int i = 0; i < size; i++)
                cout << items[i] << " ";
        }




//fills the list with a series of user element inputs
void AList::Read() {
        //Pre: none
        //Post: the Native Object AList is valid
        element userval;        //input - user input of a single element
        size = 0;               //LCV - size of array, items[]

        //Read data from user
        cout << "Enter a series of elements, " << SENTINEL
                << " to stop: ";
        userval = read_element();
        while ((size < MLS) && (userval != SENTINEL)) {
                items[size] = userval;
                size++;
                if (size >= MLS)
                        cout << "The array is full, exiting." << endl;
                else
                        userval = read_element();
                }

        //List is not known to be ordered after input
```

```
        list_ordered = false;
        }









//fills the list with a series of randomly generated elements
void AList::GenRandList(int list_size, int range_high, int range_low) {
        //Pre: none
        //Post: the Native Object AList is valid
        size = 0;       //LCV - size of array, items[]

        //create list with randomly generated values
        while (size < list_size) {
                items[size] = (rand() % (range_high - range_low + 1))
                        + range_low;
                size++;
                }

        //List is not known to be ordered after input
        list_ordered = false;
}









//swaps the elements in the position specified
void AList::Swap(element first, element second) {
        //Pre: the Native Object AList is valid
        //Post: the Native Object Alist is unchanged, except elements
                //in position [first] and [second] has swapped places
        element temp;
        temp = items[first];
        items[first] = items[second];
        items[second] = temp;
        }









//*********************************End I/O*********************************

//-------------------------------Sorts-----------------------------------------




//sorts the list using bubble sort
void AList::BubbleSort(int & comps, int & moves) {
        //Pre: the Native Object AList is valid
        //Post: the Native Object AList is unchanged, except its elements
                //are in ascending order
        comps = 0;              //Accumulator - counts # comparisons
        moves = 0;              //Accumulator - counts # moves

        for (int i = 0; i < size - 1; i++)
                for (int j = 0; j < size - 1 - i; j++) {
                        comps++;
                        if (items[j] > items[j+1]) {
```



```
                                moves += 3;
                                Swap(j, j+1);
                                }
                        else
                                ;
                        }

        //List is known to be ordered after sorting
        list_ordered = true;
        }









//sorts the list using insertion sort
void AList::InsertionSort(int & comps, int & moves) {
        //Pre: the Native Object AList is valid
        //Post: the Native Object AList is unchanged, except its elements
                //are in ascending order
        int j;          //LCV - keeps track of current element position
        bool done;      //LCV - when elements to its left are sorted, true
        comps = 0;      //Accumulator - counts # comparisons
        moves = 0;      //Accumulator - counts # moves

        for (int i = 1; i < size; i++) {
                j = i;
                done = false;
                while ( (j >= 1) && (!done) ) {
                        comps++;
                        if (items[j] < items[j-1]) {
                                moves += 3;
                                Swap (j, j-1);
                                j--;
                                }
                        else
                                done = true;
                        }
                }

        //List is known to be ordered after sorting
        list_ordered = true;
        }









//sorts the list using selection sort
void AList::SelectionSort(int & comps, int & moves) {
        //Pre: the Native Object AList is valid
        //Post: the Native Object AList is unchanged, except its elements
                //are in ascending order
        int maxpos;             //LCV - location of highest value
        comps = 0;              //Accumulator - counts # comparisons
        moves = 0;              //Accumulator - counts # moves

        for (int i = size - 1; i > 0; i--) {
                maxpos = 0;
                for (int j = 1; j <= i; j++) {
                        comps++;
                        if (items[j] > items[maxpos])
                                maxpos = j;
                        else
                                ;
                        }
```

```
                moves += 3;
                Swap(maxpos, i);
                }

        //List is known to be ordered after sorting
        list_ordered = true;
        }




//*********************************End Sorts*******************************

//--------------------------------Searches----------------------------------



//searches the list for the specified target, using linear search
void AList::LinearSearch(element target, bool & found, int & position,
        int & comps) {
        //Pre: the Native Object AList is valid and target is a valid element
        //Post: 1) if target exist on the Native Object Alist,
                //found will be true and position will be a location of the
                //target on N.O. AList
                // 2) otherwise, target will be false and position will be
                // undefined (make no promises)
        found = false;          //LCV - target not found at first
        position = 0;           //LCV - position of current position
        comps = 0;              //Accumulator - counts # comparisons

        while ( (!found) && (position < size) ) {
                comps++;
                if (items[position] == target)
                        found = true;
                else
                        position++;
                }
        }









//searches the list for the specified target, using binary search
void AList::BinarySearch(element target, bool & found, int & position,
        int & comps) {
        //Pre: the Native Object Alist is valid AND in ascending order and
                //target is a valid element
        //Post: 1) if target exist on the Native Object Alist,
                //found will be true and position will be a location of the
                //target on N.O. AList
                // 2) otherwise, target will be false and position will be
                // undefined (make no promises)
        int low;        //LCV - lowest position of "interesting" part of list
        int high;       //LCV - highest position of "interesting" part of list
        int mid;        //LCV - middle position of "interesting" part of list
        found = false;  //LCV - target not found at first
        comps = 0;      //Accumulator - counts # comparisons
        low = 0;
        high = size - 1;

        while ( (!found) && (low <= high) ) {
                mid = (low + high) / 2;
                comps++;
```



```
                if (target == items[mid]) {
                        found = true;
                        position = mid;
                        }
                else if (target < items[mid]) {
                        comps++;
                        high = mid - 1;
                        }
                else {//target > items[mid]
                        comps++;
                        low = mid + 1;
                        }
                }
        }









//*********************************End Searches*****************************

//--------------------------------Stats-------------------------------------



//calculates the theoretical computations/moves required for quadratic sorts
int AList::CalcQuad() {
        int result;     //result of theoretical quadratic comp/move

        result = (size * size / 2) - (size / 2);

        return result;
        }









//calculates the theoretical computations required for logarithmic searches
int AList::CalcLog() {
        int remain;     //LCV - size of list; size of list as it's halved
        int counter;    //Accumulator - counts # times list is halved

        remain = size;
        counter = 0;
        while (remain > 0) {
                remain /= 2;
                counter++;
                }

        return counter;
        }



//*********************************End Stats********************************

//--------------------------------Public Methods----------------------------



//should be called right after the creation of the object
//sets the N.O. AList to be a valid empty list
void AList::FirstLoad() {
        //Pre: the N.O. AList cannot be valid
        //Post: the N.O. AList is valid (specifically, AList is empty)
```

```
        size = 0;
        list_ordered = true;
        }




//runs Print()
void AList::Run_Print() {
        cout << "Current list:  ";

        //display contents of list
        if (size > 0)
                Print();
        else //size <= 0
                cout << "(empty) ";

        if (list_ordered == true)
                cout << "(KNOWN to be ordered)" << endl << endl;
        else //list_ordered == false
                cout << "(NOT KNOWN to be ordered)" << endl << endl;
        }







//runs Read(), display output
void AList::Run_Read() {
        cout << endl << "Resetting the current list from the keyboard."
                << endl << endl;

        Read();

        cout << endl << "Finished resetting, " << size
                << " elements entered." << endl << endl;
        }









//runs GenRandList(), display output
void AList::Run_GenRandList() {
        int list_size;          //input - desired list size
        int range_high;         //input - desired upper limit
        int range_low;          //input - desired lower limit

        cout << endl << "Resetting the current list "
                << "using randomly generated elements." << endl << endl;

        //get desired list size
        cout << "Enter the desired number of elements (0 to " << MLS << "): ";
        list_size = read_int();
        while ((list_size > MLS) || (list_size < 0)) {
                cout << "Response must be between 0 and " << MLS
                        << ", try again: ";
                list_size = read_int();
                }

        //get desired lower limit
        cout << "Enter the lower limit of the range: ";
        range_low = read_int();
```



```
        //get desired upper limit
        cout << "Enter the upper limit of the range: ";
        range_high = read_int();
        while (range_high < range_low) {
                cout << "Must be a value higher than " << range_low
                        << ", try again: ";
                range_high = read_int();
                }

        GenRandList(list_size, range_high, range_low);

        //display confirmation
        cout << endl << "Finished resetting, " << size
                << " elements between " << range_low
                << " and " << range_high
                << " randomly generated." << endl << endl;
}




//runs BubbleSort(), display calculations/moves required for sort
void AList::Run_BubbleSort() {
        int comps;              //Accumulator - counts # comparisons
        int moves;              //Accumulator - counts # moves

        cout << endl << "Performing Bubble Sort on the current list." << endl;

        BubbleSort(comps, moves);

        cout << endl << "Theoretical sort statistics:  " << CalcQuad()
                << " element comparisons, " << 3 * CalcQuad()
                << " element movements " << endl;

        cout << "Actual sort statistics:       " << comps
                << " element comparisons, " << moves
                << " element movements " << endl;

        cout << endl <<  "Finishing Bubble Sort." << endl << endl;
        }




//runs InsertionSort(), display calculations/moves required for sort
void AList::Run_InsertionSort() {
        int comps;              //Accumulator - counts # comparisons
        int moves;              //Accumulator - counts # moves

        cout << endl << "Performing Insertion Sort on the current list."
                << endl;

        InsertionSort(comps, moves);

        cout << endl << "Theoretical sort statistics:  " << CalcQuad()
                << " element comparisons, " << 3 * CalcQuad()
                << " element movements " << endl;

        cout << "Actual sort statistics:       " << comps
                << " element comparisons, " << moves
                << " element movements " << endl;

        cout << endl <<  "Finishing Insertion Sort." << endl << endl;
        }
```

```
//runs SelectionSort(), display calculations/moves required for sort
void AList::Run_SelectionSort() {
        int comps;              //Accumulator - counts # comparisons
        int moves;              //Accumulator - counts # moves

        cout << endl << "Performing Selection Sort on the current list."
                << endl;

        SelectionSort(comps, moves);

        cout << endl << "Theoretical sort statistics:  " << CalcQuad()
                << " element comparisons, ";

        if (size > 0)
                cout << 3 * (size - 1);
        else
                cout << 0;

        cout << " element movements " << endl;

        cout << "Actual sort statistics:       " << comps
                << " element comparisons, " << moves
                << " element movements " << endl;

        cout << endl <<  "Finishing Selection Sort." << endl << endl;
        }




//runs LinearSearch(), display output, and calculations required for search
void AList::Run_LinearSearch() {
        element target;         //input - element user wants to find
        bool found;             //LCV - target not found at first
        int position;           //LCV - position of current position
        int comps;              //Accumulator - counts # comparisons

        cout << endl << "Performing Linear Search on the current list."
                << endl << endl;

        //get desired target from user
        cout << "Enter a target element to search for: ";
        target = read_element();

        LinearSearch(target, found, position, comps);

        if (found == true)
                cout << endl << "The target was FOUND on the current list "
                        << "in position " << position << "." << endl;
        else // found == false
                cout << endl << "The target was NOT FOUND on the current list."
                        << endl;

        cout << endl << "Theoretical search statistics:  " << size
                << " element comparisons" << endl;

        cout << "Actual search statistics:       " << comps
                << " element comparisons" << endl;

        cout << endl << "Finishing Linear Search." << endl << endl;
        }
```



```
//runs BinarySearch(), display output, and calculations required for search
void AList::Run_BinarySearch() {
        //only run binary search when the list is known to be ordered

        if (list_ordered == true) {
                element target;         //input - element user wants to find
                bool found;             //LCV - target not found at first
                int position;           //LCV - position of current position
                int comps;              //Accumulator - counts # comparisons

                cout << endl << "Performing Binary Search on the current list."
                << endl << endl;

                //get desired target from user
                cout << "Enter a target element to search for: ";
                target = read_element();

                BinarySearch(target, found, position, comps);

                if (found == true)
                        cout << endl << "The target was FOUND on "
                                << "the current list in position "
                                << position << "." << endl;
                else // found == false
                        cout << endl << "The target was NOT FOUND "
                                << "on the current list." << endl;

                cout << endl << "Theoretical search statistics:  "
                        << 2 * CalcLog() << " element comparisons" << endl;

                cout << "Actual search statistics:       " << comps
                        << " element comparisons" << endl;

                cout << endl << "Finishing Binary Search." << endl << endl;
                }
        else //list_ordered == false
                cout << endl << "Sorry, since the current list is not known "
                        << "to be ordered, the Binary Search" << endl
                        << "cannot be performed at this time.  "
                        << "Please sort the current list first."
                        << endl << endl;
        }

//****************************End Public Methods************************
```

```
********************************************************************************
********************************************************************************
**                                                                            **
**                            alist.cc compilation                            **
**                                                                            **
********************************************************************************
********************************************************************************

c++ compilation succeeded
```



```
********************************************************************************
********************************************************************************
**                                                                            **
**                alist.cc execution - required testcase #1                   **
**                                                                            **
********************************************************************************
********************************************************************************

Sort and Search Demo Program, version 1.0
(c) 2011, (Steven Liu)

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 2

Resetting the current list using randomly generated elements.

Enter the desired number of elements (0 to 50): 10
Enter the lower limit of the range: 1
Enter the upper limit of the range: 25

Finished resetting, 10 elements between 1 and 25 randomly generated.

Current list:  3 9 18 7 14 20 1 23 21 6 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 3

Performing Bubble Sort on the current list.

Theoretical sort statistics:  45 element comparisons, 135 element movements
Actual sort statistics:       45 element comparisons, 51 element movements

Finishing Bubble Sort.

Current list:  1 3 6 7 9 14 18 20 21 23 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 2

Resetting the current list using randomly generated elements.
```

```
Enter the desired number of elements (0 to 50): 20
Enter the lower limit of the range: 10
Enter the upper limit of the range: 30

Finished resetting, 20 elements between 10 and 30 randomly generated.

Current list:  26 26 24 30 19 21 28 19 22 11 20 11 21 23 19 13 18 10 27 13 (NOT
KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 4

Performing Insertion Sort on the current list.

Theoretical sort statistics:  190 element comparisons, 570 element movements
Actual sort statistics:       144 element comparisons, 387 element movements

Finishing Insertion Sort.

Current list:  10 11 11 13 13 18 19 19 19 20 21 21 22 23 24 26 26 27 28 30 (KNOW
N to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 2

Resetting the current list using randomly generated elements.

Enter the desired number of elements (0 to 50): 30
Enter the lower limit of the range: 10
Enter the upper limit of the range: 20

Finished resetting, 30 elements between 10 and 20 randomly generated.

Current list:  15 14 13 12 13 20 10 11 16 10 20 16 11 12 20 19 14 19 19 15 10 14
 16 16 12 19 16 12 12 15 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 5

Performing Selection Sort on the current list.
```



```
Theoretical sort statistics:  435 element comparisons, 87 element movements
Actual sort statistics:       435 element comparisons, 87 element movements

Finishing Selection Sort.

Current list:  10 10 10 11 11 12 12 12 12 12 13 13 14 14 14 15 15 15 16 16 16 16
 16 19 19 19 19 20 20 20 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 1

Resetting the current list from the keyboard.

Enter a series of elements, −1 to stop: 10 20 15 5 25 −1

Finished resetting, 5 elements entered.

Current list:  10 20 15 5 25 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 6

Performing Linear Search on the current list.

Enter a target element to search for: 15

The target was FOUND on the current list in position 2.

Theoretical search statistics:  5 element comparisons
Actual search statistics:       3 element comparisons

Finishing Linear Search.

Current list:  10 20 15 5 25 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 6

Performing Linear Search on the current list.

Enter a target element to search for: 16
```

```
The target was NOT FOUND on the current list.

Theoretical search statistics:  5 element comparisons
Actual search statistics:       5 element comparisons

Finishing Linear Search.

Current list:  10 20 15 5 25 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 3

Performing Bubble Sort on the current list.

Theoretical sort statistics:  10 element comparisons, 30 element movements
Actual sort statistics:       10 element comparisons, 12 element movements

Finishing Bubble Sort.

Current list:  5 10 15 20 25 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 7

Performing Binary Search on the current list.

Enter a target element to search for: 5

The target was FOUND on the current list in position 0.

Theoretical search statistics:  6 element comparisons
Actual search statistics:       3 element comparisons

Finishing Binary Search.

Current list:  5 10 15 20 25 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 7

Performing Binary Search on the current list.
```



```
Enter a target element to search for: 6

The target was NOT FOUND on the current list.

Theoretical search statistics:  6 element comparisons
Actual search statistics:       6 element comparisons

Finishing Binary Search.

Current list:  5 10 15 20 25 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 8

Quitting Sort and Search Demo Program, version 1.0
```

```
********************************************************************************
********************************************************************************
**                                                                            **
**                 alist.cc execution - required testcase #2                  **
**                                                                            **
********************************************************************************
********************************************************************************

Sort and Search Demo Program, version 1.0
(c) 2011, (Steven Liu)

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 3

Performing Bubble Sort on the current list.

Theoretical sort statistics:  0 element comparisons, 0 element movements
Actual sort statistics:       0 element comparisons, 0 element movements

Finishing Bubble Sort.

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 4

Performing Insertion Sort on the current list.

Theoretical sort statistics:  0 element comparisons, 0 element movements
Actual sort statistics:       0 element comparisons, 0 element movements

Finishing Insertion Sort.

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 5

Performing Selection Sort on the current list.
```



```
Theoretical sort statistics:  0 element comparisons, 0 element movements
Actual sort statistics:       0 element comparisons, 0 element movements

Finishing Selection Sort.

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 6

Performing Linear Search on the current list.

Enter a target element to search for: 1

The target was NOT FOUND on the current list.

Theoretical search statistics:  0 element comparisons
Actual search statistics:       0 element comparisons

Finishing Linear Search.

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 7

Performing Binary Search on the current list.

Enter a target element to search for: 1

The target was NOT FOUND on the current list.

Theoretical search statistics:  0 element comparisons
Actual search statistics:       0 element comparisons

Finishing Binary Search.

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 8
```

```
Quitting Sort and Search Demo Program, version 1.0
```



```
********************************************************************************
********************************************************************************
**                                                                            **
**                 alist.cc execution - required testcase #3                  **
**                                                                            **
********************************************************************************
********************************************************************************

Sort and Search Demo Program, version 1.0
(c) 2011, (Steven Liu)

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 1

Resetting the current list from the keyboard.

Enter a series of elements, -1 to stop: maybe
Bad input datatype; Try again: possibly probably
Bad input datatype; Try again: improbably
Bad input datatype; Try again: -1

Finished resetting, 0 elements entered.

Current list:  (empty) (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 2

Resetting the current list using randomly generated elements.

Enter the desired number of elements (0 to 50): purple
Response must be a whole number, try again: 50
Enter the lower limit of the range: yellow
Response must be a whole number, try again: 0
Enter the upper limit of the range: green
Response must be a whole number, try again: 99

Finished resetting, 50 elements between 0 and 99 randomly generated.

Current list:  43 45 67 55 60 96 17 15 82 96 43 26 55 83 19 98 94 57 22 8 93 72
88 53 54 99 28 18 29 47 11 72 44 30 79 5 79 97 20 61 93 63 40 0 46 11 99 92 68 2
1 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
```

```
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 2

Resetting the current list using randomly generated elements.

Enter the desired number of elements (0 to 50): −4
Response must be between 0 and 50, try again: 10
Enter the lower limit of the range: 10
Enter the upper limit of the range: 5
Must be a value higher than 10, try again: 8
Must be a value higher than 10, try again: 9
Must be a value higher than 10, try again: 10

Finished resetting, 10 elements between 10 and 10 randomly generated.

Current list:  10 10 10 10 10 10 10 10 10 10 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 1

Resetting the current list from the keyboard.

Enter a series of elements, −1 to stop: −100
−1

Finished resetting, 1 elements entered.

Current list:  −100 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 2

Resetting the current list using randomly generated elements.

Enter the desired number of elements (0 to 50): 5
Enter the lower limit of the range: 1
Enter the upper limit of the range: 10

Finished resetting, 5 elements between 1 and 10 randomly generated.

Current list:  5 4 7 10 4 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
```



```
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 7

Sorry, since the current list is not known to be ordered, the Binary Search
cannot be performed at this time.  Please sort the current list first.

Current list:  5 4 7 10 4 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: yes
Response must be a whole number, try again: 8

Quitting Sort and Search Demo Program, version 1.0
```

```
********************************************************************************
********************************************************************************
**                                                                            **
**    alist.cc execution − proposed normal testcase Normal testcases [#1]     **
**                                                                            **
********************************************************************************
********************************************************************************

Sort and Search Demo Program, version 1.0
(c) 2011, (Steven Liu)

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 1

Resetting the current list from the keyboard.

Enter a series of elements, −1 to stop: 1 5 10 9 5 45 3 −10 −5 −4 −1

Finished resetting, 10 elements entered.

Current list:  1 5 10 9 5 45 3 −10 −5 −4 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 6

Performing Linear Search on the current list.

Enter a target element to search for: −4

The target was FOUND on the current list in position 9.

Theoretical search statistics:  10 element comparisons
Actual search statistics:       10 element comparisons

Finishing Linear Search.

Current list:  1 5 10 9 5 45 3 −10 −5 −4 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 3
```



```
Performing Bubble Sort on the current list.

Theoretical sort statistics:  45 element comparisons, 135 element movements
Actual sort statistics:       45 element comparisons, 87 element movements

Finishing Bubble Sort.

Current list:  −10 −5 −4 1 3 5 5 9 10 45 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 4

Performing Insertion Sort on the current list.

Theoretical sort statistics:  45 element comparisons, 135 element movements
Actual sort statistics:       9 element comparisons, 0 element movements

Finishing Insertion Sort.

Current list:  −10 −5 −4 1 3 5 5 9 10 45 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 2

Resetting the current list using randomly generated elements.

Enter the desired number of elements (0 to 50): 25
Enter the lower limit of the range: 10
Enter the upper limit of the range: 100

Finished resetting, 25 elements between 10 and 100 randomly generated.

Current list:  22 74 45 38 99 38 88 51 62 28 65 66 27 21 88 71 79 23 66 73 20 84
 79 17 81 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 5

Performing Selection Sort on the current list.

Theoretical sort statistics:  300 element comparisons, 72 element movements
```

```
Actual sort statistics:       300 element comparisons, 72 element movements

Finishing Selection Sort.

Current list:  17 20 21 22 23 27 28 38 38 45 51 62 65 66 66 71 73 74 79 79 81 84
 88 88 99 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 5

Performing Selection Sort on the current list.

Theoretical sort statistics:  300 element comparisons, 72 element movements
Actual sort statistics:       300 element comparisons, 72 element movements

Finishing Selection Sort.

Current list:  17 20 21 22 23 27 28 38 38 45 51 62 65 66 66 71 73 74 79 79 81 84
 88 88 99 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 8

Quitting Sort and Search Demo Program, version 1.0
```



```
********************************************************************************
********************************************************************************
**                                                                            **
**           alist.cc execution - proposed boundary testcase 1 [#1]           **
**                                                                            **
********************************************************************************
********************************************************************************

Sort and Search Demo Program, version 1.0
(c) 2011, (Steven Liu)

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 1

Resetting the current list from the keyboard.

Enter a series of elements, -1 to stop: 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0
1
2
3
4
5
6
7
8
9
0
1
23
3
4
5
6
7
8
9
The array is full, exiting.

Finished resetting, 50 elements entered.

Current list:  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 6

Performing Linear Search on the current list.
```

```
Enter a target element to search for: 0

The target was FOUND on the current list in position 0.

Theoretical search statistics:  50 element comparisons
Actual search statistics:       1 element comparisons

Finishing Linear Search.

Current list:  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 6

Performing Linear Search on the current list.

Enter a target element to search for: 9

The target was FOUND on the current list in position 9.

Theoretical search statistics:  50 element comparisons
Actual search statistics:       10 element comparisons

Finishing Linear Search.

Current list:  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 5

Performing Selection Sort on the current list.

Theoretical sort statistics:  1225 element comparisons, 147 element movements
Actual sort statistics:       1225 element comparisons, 147 element movements

Finishing Selection Sort.

Current list:  0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 6 6 6
 6 6 7 7 7 7 7 8 8 8 8 8 9 9 9 9 9 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
```



```
   8.  Quit the program

Choose an action: 4

Performing Insertion Sort on the current list.

Theoretical sort statistics:  1225 element comparisons, 3675 element movements
Actual sort statistics:       49 element comparisons, 0 element movements

Finishing Insertion Sort.

Current list:  0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 6 6 6
 6 6 7 7 7 7 7 8 8 8 8 8 9 9 9 9 9 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 3

Performing Bubble Sort on the current list.

Theoretical sort statistics:  1225 element comparisons, 3675 element movements
Actual sort statistics:       1225 element comparisons, 0 element movements

Finishing Bubble Sort.

Current list:  0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 6 6 6
 6 6 7 7 7 7 7 8 8 8 8 8 9 9 9 9 9 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 6

Performing Linear Search on the current list.

Enter a target element to search for: 0

The target was FOUND on the current list in position 0.

Theoretical search statistics:  50 element comparisons
Actual search statistics:       1 element comparisons

Finishing Linear Search.

Current list:  0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 6 6 6
 6 6 7 7 7 7 7 8 8 8 8 8 9 9 9 9 9 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
```

```
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 6

Performing Linear Search on the current list.

Enter a target element to search for: 9

The target was FOUND on the current list in position 45.

Theoretical search statistics:  50 element comparisons
Actual search statistics:       46 element comparisons

Finishing Linear Search.

Current list:  0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 6 6 6
 6 6 7 7 7 7 7 8 8 8 8 8 9 9 9 9 9 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 2

Resetting the current list using randomly generated elements.

Enter the desired number of elements (0 to 50): 0
Enter the lower limit of the range: 0
Enter the upper limit of the range: 1

Finished resetting, 0 elements between 0 and 1 randomly generated.

Current list:  (empty) (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 2

Resetting the current list using randomly generated elements.

Enter the desired number of elements (0 to 50): 50
Enter the lower limit of the range: −50
Enter the upper limit of the range: 50

Finished resetting, 50 elements between −50 and 50 randomly generated.

Current list:  −49 35 −39 −19 24 43 −11 49 2 −44 −31 −50 25 −20 41 −38 26 41 19
−38 −18 −45 14 −24 42 −7 15 46 −37 35 −50 −36 19 −38 −39 −7 21 0 −43 −28 −28 −23
 39 13 −27 −5 25 49 −15 −6 (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
```



```
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 5

Performing Selection Sort on the current list.

Theoretical sort statistics:  1225 element comparisons, 147 element movements
Actual sort statistics:       1225 element comparisons, 147 element movements

Finishing Selection Sort.

Current list:  −50 −50 −49 −45 −44 −43 −39 −39 −38 −38 −38 −37 −36 −31 −28 −28 −
27 −24 −23 −20 −19 −18 −15 −11 −7 −7 −6 −5 0 2 13 14 15 19 19 21 24 25 25 26 35
35 39 41 41 42 43 46 49 49 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 7

Performing Binary Search on the current list.

Enter a target element to search for: −50

The target was FOUND on the current list in position 0.

Theoretical search statistics:  12 element comparisons
Actual search statistics:       9 element comparisons

Finishing Binary Search.

Current list:  −50 −50 −49 −45 −44 −43 −39 −39 −38 −38 −38 −37 −36 −31 −28 −28 −
27 −24 −23 −20 −19 −18 −15 −11 −7 −7 −6 −5 0 2 13 14 15 19 19 21 24 25 25 26 35
35 39 41 41 42 43 46 49 49 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 7

Performing Binary Search on the current list.

Enter a target element to search for: 49

The target was FOUND on the current list in position 48.

Theoretical search statistics:  12 element comparisons
Actual search statistics:       9 element comparisons

Finishing Binary Search.
```

```
Current list:  −50 −50 −49 −45 −44 −43 −39 −39 −38 −38 −38 −37 −36 −31 −28 −28 −
27 −24 −23 −20 −19 −18 −15 −11 −7 −7 −6 −5 0 2 13 14 15 19 19 21 24 25 25 26 35
35 39 41 41 42 43 46 49 49 (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 2

Resetting the current list using randomly generated elements.

Enter the desired number of elements (0 to 50): 50
Enter the lower limit of the range: −999
Enter the upper limit of the range: 999

Finished resetting, 50 elements between −999 and 999 randomly generated.

Current list:  323 −747 341 710 98 722 −2 −586 −511 8 325 −550 964 −356 −83 −835
 −741 622 −609 977 420 600 −826 −440 431 −349 905 236 229 −728 −913 −447 −475 50
1 −664 622 223 407 −891 711 −585 433 −766 −620 −850 151 −383 −592 −154 80 (NOT K
NOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 5

Performing Selection Sort on the current list.

Theoretical sort statistics:  1225 element comparisons, 147 element movements
Actual sort statistics:       1225 element comparisons, 147 element movements

Finishing Selection Sort.

Current list:  −913 −891 −850 −835 −826 −766 −747 −741 −728 −664 −620 −609 −592
−586 −585 −550 −511 −475 −447 −440 −383 −356 −349 −154 −83 −2 8 80 98 151 223 22
9 236 323 325 341 407 420 431 433 501 600 622 622 710 711 722 905 964 977 (KNOWN
 to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 7

Performing Binary Search on the current list.

Enter a target element to search for: −913
```



```
The target was FOUND on the current list in position 0.

Theoretical search statistics:  12 element comparisons
Actual search statistics:       9 element comparisons

Finishing Binary Search.

Current list:  −913 −891 −850 −835 −826 −766 −747 −741 −728 −664 −620 −609 −592
−586 −585 −550 −511 −475 −447 −440 −383 −356 −349 −154 −83 −2 8 80 98 151 223 22
9 236 323 325 341 407 420 431 433 501 600 622 622 710 711 722 905 964 977 (KNOWN
 to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 7

Performing Binary Search on the current list.

Enter a target element to search for: 977

The target was FOUND on the current list in position 49.

Theoretical search statistics:  12 element comparisons
Actual search statistics:       11 element comparisons

Finishing Binary Search.

Current list:  −913 −891 −850 −835 −826 −766 −747 −741 −728 −664 −620 −609 −592
−586 −585 −550 −511 −475 −447 −440 −383 −356 −349 −154 −83 −2 8 80 98 151 223 22
9 236 323 325 341 407 420 431 433 501 600 622 622 710 711 722 905 964 977 (KNOWN
 to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 6

Performing Linear Search on the current list.

Enter a target element to search for: −913

The target was FOUND on the current list in position 0.

Theoretical search statistics:  50 element comparisons
Actual search statistics:       1 element comparisons

Finishing Linear Search.

Current list:  −913 −891 −850 −835 −826 −766 −747 −741 −728 −664 −620 −609 −592
−586 −585 −550 −511 −475 −447 −440 −383 −356 −349 −154 −83 −2 8 80 98 151 223 22
9 236 323 325 341 407 420 431 433 501 600 622 622 710 711 722 905 964 977 (KNOWN
 to be ordered)

Actions:
```

```
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 6

Performing Linear Search on the current list.

Enter a target element to search for: 977

The target was FOUND on the current list in position 49.

Theoretical search statistics:  50 element comparisons
Actual search statistics:       50 element comparisons

Finishing Linear Search.

Current list:  −913 −891 −850 −835 −826 −766 −747 −741 −728 −664 −620 −609 −592
−586 −585 −550 −511 −475 −447 −440 −383 −356 −349 −154 −83 −2 8 80 98 151 223 22
9 236 323 325 341 407 420 431 433 501 600 622 622 710 711 722 905 964 977 (KNOWN
 to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 8

Quitting Sort and Search Demo Program, version 1.0
```



```
********************************************************************************
********************************************************************************
**                                                                            **
** alist.cc execution − proposed exception testcase Exception testcases [#1]  **
**                                                                            **
********************************************************************************
********************************************************************************

Sort and Search Demo Program, version 1.0
(c) 2011, (Steven Liu)

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 0

Invalid choice; Please choosebetween menu options 1−8

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: −1

Invalid choice; Please choosebetween menu options 1−8

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 9

Invalid choice; Please choosebetween menu options 1−8

Current list:  (empty) (KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
```

```
   8.  Quit the program

Choose an action: hello world
Response must be a whole number, try again: 1

Resetting the current list from the keyboard.

Enter a series of elements, −1 to stop: abc
Bad input datatype; Try again: two
Bad input datatype; Try again: badinput
Bad input datatype; Try again: .2187
Bad input datatype; Try again: .1
Bad input datatype; Try again: −.1
Bad input datatype; Try again: −1

Finished resetting, 0 elements entered.

Current list:  (empty) (NOT KNOWN to be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 2

Resetting the current list using randomly generated elements.

Enter the desired number of elements (0 to 50): 60
Response must be between 0 and 50, try again: 51
Response must be between 0 and 50, try again: −1
Response must be between 0 and 50, try again: −50
Response must be between 0 and 50, try again: 50.1
Enter the lower limit of the range: Response must be a whole number, try again:
−999
Enter the upper limit of the range: 999999999999
Response must be a whole number, try again: 99999

Finished resetting, 50 elements between −999 and 99999 randomly generated.

Current list:  5211 80085 2970 93880 56601 99803 41324 22058 59011 79822 2896 49
189 87155 79060 91262 61459 63645 10812 −968 56342 68244 43009 66238 15771 7761
15840 63371 96210 44507 33778 29239 50717 13863 33208 1687 71464 33011 44010 516
11 93022 81921 12596 −699 26167 92655 48653 88625 13391 17554 45746 (NOT KNOWN t
o be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 6

Performing Linear Search on the current list.

Enter a target element to search for: onetwothree
Bad input datatype; Try again: space
Bad input datatype; Try again: 99ll
```



```
The target was NOT FOUND on the current list.

Theoretical search statistics:  50 element comparisons
Actual search statistics:       50 element comparisons

Finishing Linear Search.

Current list:  5211 80085 2970 93880 56601 99803 41324 22058 59011 79822 2896 49
189 87155 79060 91262 61459 63645 10812 −968 56342 68244 43009 66238 15771 7761
15840 63371 96210 44507 33778 29239 50717 13863 33208 1687 71464 33011 44010 516
11 93022 81921 12596 −699 26167 92655 48653 88625 13391 17554 45746 (NOT KNOWN t
o be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: Response must be a whole number, try again: 7

Sorry, since the current list is not known to be ordered, the Binary Search
cannot be performed at this time.  Please sort the current list first.

Current list:  5211 80085 2970 93880 56601 99803 41324 22058 59011 79822 2896 49
189 87155 79060 91262 61459 63645 10812 −968 56342 68244 43009 66238 15771 7761
15840 63371 96210 44507 33778 29239 50717 13863 33208 1687 71464 33011 44010 516
11 93022 81921 12596 −699 26167 92655 48653 88625 13391 17554 45746 (NOT KNOWN t
o be ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 5

Performing Selection Sort on the current list.

Theoretical sort statistics:  1225 element comparisons, 147 element movements
Actual sort statistics:       1225 element comparisons, 147 element movements

Finishing Selection Sort.

Current list:  −968 −699 1687 2896 2970 5211 7761 10812 12596 13391 13863 15771
15840 17554 22058 26167 29239 33011 33208 33778 41324 43009 44010 44507 45746 48
653 49189 50717 51611 56342 56601 59011 61459 63371 63645 66238 68244 71464 7906
0 79822 80085 81921 87155 88625 91262 92655 93022 93880 96210 99803 (KNOWN to be
 ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program
```

```
Choose an action: 7

Performing Binary Search on the current list.

Enter a target element to search for: blahblah
Bad input datatype; Try again: 999999

The target was NOT FOUND on the current list.

Theoretical search statistics:  12 element comparisons
Actual search statistics:       12 element comparisons

Finishing Binary Search.

Current list:  −968 −699 1687 2896 2970 5211 7761 10812 12596 13391 13863 15771
15840 17554 22058 26167 29239 33011 33208 33778 41324 43009 44010 44507 45746 48
653 49189 50717 51611 56342 56601 59011 61459 63371 63645 66238 68244 71464 7906
0 79822 80085 81921 87155 88625 91262 92655 93022 93880 96210 99803 (KNOWN to be
 ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 6

Performing Linear Search on the current list.

Enter a target element to search for: −9999

The target was NOT FOUND on the current list.

Theoretical search statistics:  50 element comparisons
Actual search statistics:       50 element comparisons

Finishing Linear Search.

Current list:  −968 −699 1687 2896 2970 5211 7761 10812 12596 13391 13863 15771
15840 17554 22058 26167 29239 33011 33208 33778 41324 43009 44010 44507 45746 48
653 49189 50717 51611 56342 56601 59011 61459 63371 63645 66238 68244 71464 7906
0 79822 80085 81921 87155 88625 91262 92655 93022 93880 96210 99803 (KNOWN to be
 ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 6

Performing Linear Search on the current list.

Enter a target element to search for: 999999

The target was NOT FOUND on the current list.

Theoretical search statistics:  50 element comparisons
Actual search statistics:       50 element comparisons
```



```
Finishing Linear Search.

Current list:  −968 −699 1687 2896 2970 5211 7761 10812 12596 13391 13863 15771
15840 17554 22058 26167 29239 33011 33208 33778 41324 43009 44010 44507 45746 48
653 49189 50717 51611 56342 56601 59011 61459 63371 63645 66238 68244 71464 7906
0 79822 80085 81921 87155 88625 91262 92655 93022 93880 96210 99803 (KNOWN to be
 ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 7

Performing Binary Search on the current list.

Enter a target element to search for: −99999

The target was NOT FOUND on the current list.

Theoretical search statistics:  12 element comparisons
Actual search statistics:       10 element comparisons

Finishing Binary Search.

Current list:  −968 −699 1687 2896 2970 5211 7761 10812 12596 13391 13863 15771
15840 17554 22058 26167 29239 33011 33208 33778 41324 43009 44010 44507 45746 48
653 49189 50717 51611 56342 56601 59011 61459 63371 63645 66238 68244 71464 7906
0 79822 80085 81921 87155 88625 91262 92655 93022 93880 96210 99803 (KNOWN to be
 ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: byebye
Response must be a whole number, try again: 100

Invalid choice; Please choosebetween menu options 1−8

Current list:  −968 −699 1687 2896 2970 5211 7761 10812 12596 13391 13863 15771
15840 17554 22058 26167 29239 33011 33208 33778 41324 43009 44010 44507 45746 48
653 49189 50717 51611 56342 56601 59011 61459 63371 63645 66238 68244 71464 7906
0 79822 80085 81921 87155 88625 91262 92655 93022 93880 96210 99803 (KNOWN to be
 ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program
```

```
Choose an action: 9

Invalid choice; Please choosebetween menu options 1−8

Current list:  −968 −699 1687 2896 2970 5211 7761 10812 12596 13391 13863 15771
15840 17554 22058 26167 29239 33011 33208 33778 41324 43009 44010 44507 45746 48
653 49189 50717 51611 56342 56601 59011 61459 63371 63645 66238 68244 71464 7906
0 79822 80085 81921 87155 88625 91262 92655 93022 93880 96210 99803 (KNOWN to be
 ordered)

Actions:
   1.  Reset the current list from the keyboard
   2.  Reset the current list using randomly generated elements
   3.  Perform Bubble Sort on the current list
   4.  Perform Insertion Sort on the current list
   5.  Perform Selection Sort on the current list
   6.  Perform Linear Search on the current list
   7.  Perform Binary Search on the current list
   8.  Quit the program

Choose an action: 8

Quitting Sort and Search Demo Program, version 1.0
```