```
*******************************************************************************
*******************************************************************************
**                                                                           **
**                            extra3.cc listing                              **
**                                                                           **
*******************************************************************************
*******************************************************************************

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

/*
        Steven Liu
        CS215-J001
        Spring, 2011
        Extra Credit - Improved Binary Search
*/


//global section:
const int MLS = 1000;           //size of array
typedef int element;            //datatype of "element"



//global function prototypes:
element read_element();
//array list class prototype:
class AList {
        private:
                element items[MLS];
                int size;
                bool list_ordered;
                //-I/O:
                void Swap(element first, element second);
                //-Searches:
                void BinarySearch(element target, bool & found,
                        int & position, int & comps);
                void BinarySearchImp(element target, bool & found,
                        int & position, int & comps);
                //-Stats:
                int CalcLog();
        public:
                void FirstLoad();
                void GenRandList(int list_size, int range_high, int range_low);
                void BubbleSort();
                void Run_BinarySearch();
        };



//****main function****
int main() {
        srand(int(time(0)));    //seed the random number generator
        int high = 500;
        int low = 0;

        AList myAList;          //create object
        myAList.FirstLoad();    //prepares object for use

        myAList.GenRandList(MLS, high, low);
        myAList.BubbleSort();

        element menu_input;
        do {
                cout << endl << "There are " << MLS << " elements in the list"
                        << endl << "The list is sorted" << endl
```



```
                        << "Highest possible number in list is " << high << endl
                        << "Lowest possible number in list is " << low << endl;

                myAList.Run_BinarySearch();

                cout << "Enter 1 to continue looping, "
                        << "any other integer to stop: ";
                menu_input = read_element();
                } while (menu_input == 1);

        }




//------------------------------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;
        }




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

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







//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
        int randnum;
        while (size < list_size) {
                randnum = (rand() % (range_high - range_low + 1)) + range_low;
                items[size] = randnum;
                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() {
        //Pre: the Native Object AList is valid
        //Post: the Native Object AList is unchanged, except its elements
                //are in ascending order

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

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




//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;
                        }
                }
        }



//searches the list for the specified target, using binary search
void AList::BinarySearchImp(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]) {
                        high = mid - 1;
                        }
                else if (target > items[mid]){
                        comps++;
                        low = mid + 1;
                        }
                else { //target == items[mid]
                        comps++;
                        found = true;
                        position = mid;
                        }
                }
        }




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

```
//--------------------------------Stats----------------------------------



//calculates the 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 = 1;
        while (remain / 2 > 0) {
                remain /= 2;
                counter++;
                }

        return counter;
        }



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

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



//should be called at the creation of the object
//sets the N.O. AList to be a valid 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 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

                int comps2;             //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);
                BinarySearchImp(target, found, position, comps2);


                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 << "NORMAL actual search statistics:       " << comps
                        << " element comparisons" << endl;

                cout << "IMPROVED actual search statistics:     " << comps2
                        << " element comparisons" << endl;
                }
        else //list_ordered == false
                cout << endl << "List not ordered" << endl << endl;
        }


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

```
********************************************************************************
********************************************************************************
**                                                                            **
**                          extra3.cc compilation                             **
**                                                                            **
********************************************************************************
********************************************************************************

c++ compilation succeeded
```



```
********************************************************************************
********************************************************************************
**                                                                            **
**    extra3.cc execution - unstructured testcase out of range tests [#1]     **
**                                                                            **
********************************************************************************
********************************************************************************


There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: -0

The target was NOT FOUND on the current list.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       18 element comparisons
IMPROVED actual search statistics:     9 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 501

The target was NOT FOUND on the current list.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       20 element comparisons
IMPROVED actual search statistics:     20 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: -123

The target was NOT FOUND on the current list.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       18 element comparisons
IMPROVED actual search statistics:     9 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 1000

The target was NOT FOUND on the current list.

Theoretical search statistics:         20 element comparisons
```

```
NORMAL actual search statistics:       20 element comparisons
IMPROVED actual search statistics:     20 element comparisons
Enter 1 to continue looping, any other integer to stop: 0
```



```
********************************************************************************
********************************************************************************
**                                                                            **
**   extra3.cc execution - unstructured testcase random number choices [#2]   **
**                                                                            **
********************************************************************************
********************************************************************************


There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 123

The target was FOUND on the current list in position 252.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       15 element comparisons
IMPROVED actual search statistics:     10 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 9832

The target was NOT FOUND on the current list.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       20 element comparisons
IMPROVED actual search statistics:     20 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 84

The target was FOUND on the current list in position 158.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       15 element comparisons
IMPROVED actual search statistics:     11 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 452

The target was FOUND on the current list in position 899.

Theoretical search statistics:         20 element comparisons
```

```
NORMAL actual search statistics:       17 element comparisons
IMPROVED actual search statistics:     15 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 302

The target was NOT FOUND on the current list.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       20 element comparisons
IMPROVED actual search statistics:     15 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 4

The target was FOUND on the current list in position 8.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       17 element comparisons
IMPROVED actual search statistics:     11 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 234

The target was FOUND on the current list in position 459.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       13 element comparisons
IMPROVED actual search statistics:     12 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 299

The target was FOUND on the current list in position 579.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       19 element comparisons
IMPROVED actual search statistics:     15 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
```



```
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 39

The target was FOUND on the current list in position 72.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       15 element comparisons
IMPROVED actual search statistics:     11 element comparisons
Enter 1 to continue looping, any other integer to stop: 0
```

```
********************************************************************************
********************************************************************************
**                                                                            **
**       extra3.cc execution - unstructured testcase exact center [#3]        **
**                                                                            **
********************************************************************************
********************************************************************************


There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 250

The target was FOUND on the current list in position 518.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       15 element comparisons
IMPROVED actual search statistics:     11 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 249

The target was NOT FOUND on the current list.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       20 element comparisons
IMPROVED actual search statistics:     13 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 248

The target was FOUND on the current list in position 516.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       17 element comparisons
IMPROVED actual search statistics:     12 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 246

The target was FOUND on the current list in position 513.

Theoretical search statistics:         20 element comparisons
```



```
NORMAL actual search statistics:       19 element comparisons
IMPROVED actual search statistics:     15 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 244

The target was FOUND on the current list in position 510.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       15 element comparisons
IMPROVED actual search statistics:     11 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 242

The target was FOUND on the current list in position 506.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       13 element comparisons
IMPROVED actual search statistics:     9 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 240

The target was FOUND on the current list in position 501.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       19 element comparisons
IMPROVED actual search statistics:     13 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 239

The target was FOUND on the current list in position 499.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       1 element comparisons
IMPROVED actual search statistics:     2 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
```

```
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 238

The target was FOUND on the current list in position 497.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       17 element comparisons
IMPROVED actual search statistics:     17 element comparisons
Enter 1 to continue looping, any other integer to stop: 0
```



```
********************************************************************************
********************************************************************************
**                                                                            **
**         extra3.cc execution - unstructured testcase boundary [#4]          **
**                                                                            **
********************************************************************************
********************************************************************************


There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 0

The target was NOT FOUND on the current list.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       18 element comparisons
IMPROVED actual search statistics:     9 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 1

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

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       15 element comparisons
IMPROVED actual search statistics:     9 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 2

The target was FOUND on the current list in position 4.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       17 element comparisons
IMPROVED actual search statistics:     11 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 500

The target was FOUND on the current list in position 996.

Theoretical search statistics:         20 element comparisons
```

```
NORMAL actual search statistics:       15 element comparisons
IMPROVED actual search statistics:     16 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 499

The target was FOUND on the current list in position 992.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       13 element comparisons
IMPROVED actual search statistics:     14 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 498

The target was FOUND on the current list in position 990.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       17 element comparisons
IMPROVED actual search statistics:     17 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 497

The target was FOUND on the current list in position 988.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       15 element comparisons
IMPROVED actual search statistics:     15 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 496

The target was FOUND on the current list in position 986.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       17 element comparisons
IMPROVED actual search statistics:     16 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
```



```
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 4

The target was FOUND on the current list in position 7.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       19 element comparisons
IMPROVED actual search statistics:     12 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

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 8.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       17 element comparisons
IMPROVED actual search statistics:     11 element comparisons
Enter 1 to continue looping, any other integer to stop: 1

There are 1000 elements in the list
The list is sorted
Highest possible number in list is 500
Lowest possible number in list is 0

Performing Binary Search on the current list.

Enter a target element to search for: 1000

The target was NOT FOUND on the current list.

Theoretical search statistics:         20 element comparisons
NORMAL actual search statistics:       20 element comparisons
IMPROVED actual search statistics:     20 element comparisons
Enter 1 to continue looping, any other integer to stop: 0
```