55 pts

Name1: Andrew Gharios

Name2: _____

Class Day / Time: CS1B M-TH 5-7:20p

Due Date: 6/23/21

Lab #5: Binary Search

In this lab, you will perform the tasks bellow. DO NOT USE GLOBAL CONSTANTS!

1. Create a header file that contains the following.
   - All necessary pre-processor directives
   - The prototype for a function that sorts an array using an insertion sort.
   - The prototype for a function that searches an array using a sequential search and returns the appropriate index in the array.
   - The prototype for a function that searches an array using a binary search and returns the appropriate index in the array.
   - The prototype for a function that outputs an array.

2. Create your source files as follows:
   - Create a source file that contains the code for the search functions.
   - Create a source file that contains the code for the sort function.
   - Create a source file that contains the code for the output function.

3. Create a file that contains the main function which should perform the following tasks in order.
   - Call the output function.
   - Allow the user to input a key
   - Call the function that performs a sequential search 4 times.
     Output the index # that represents where the item was found.
   - Call the function that performs the insertion sort.
   - Call the output function.
   - Call the function that performs the binary search 4 times.
     Output the index # that represents where the item was found.

Use the following Array:
int intArray[8] = {4, 1, 7, 12, 8, 13, 9, 21};

Turn in as a single PDF file (IN THIS ORDER)

1 – The first page of this lab (fill in the information on the top right)
2 – Program output (cut and pasted into a text file within eclipse)
3 – Header file
4 – Main.cpp
5 – Search functions source file, sort function source file and output source file

```
1  ************************************************************
2  *   PROGRAMMED BY : Andrew Gharios
3  *   STUDENT ID    : 1449366
4  *   CLASS         : M-Th 5-7:20p
5  *   LAB #5        : Binary Search
6  ************************************************************
7  Index #0: 4
8  Index #1: 1
9  Index #2: 7
10 Index #3: 12
11 Index #4: 7
12 Index #5: 13
13 Index #6: 9
14 Index #7: 21
15
16 Enter an integer to search for: 9
17 The integer 9 was found in index #6.
18
19 Enter an integer to search for: 6
20 6 was not found!
21
22 Enter an integer to search for: 21
23 The integer 21 was found in index #7.
24
25 Enter an integer to search for: 4
26 The integer 4 was found in index #0.
27
28
29 Performing Insertion Sort!
30
31 Index #0: 1
32 Index #1: 4
33 Index #2: 7
34 Index #3: 7
35 Index #4: 9
36 Index #5: 12
37 Index #6: 13
38 Index #7: 21
39
40 Enter an integer to search for: 12
41 The integer 12 was found in index #5.
42
43 Enter an integer to search for: 21
44 The integer 21 was found in index #7.
45
46 Enter an integer to search for: 2
47 2 was not found!
48
49 Enter an integer to search for: 1
```

```
50  The integer 1 was found in index #0.
```

```
/*****************************************************************************
* AUTHOR     : Andrew Gharios
* STUDENT ID : 1449366
* LAB #5     : Binary Search
* CLASS      : CS1B
* SECTION    : M-TH: 5-7:20p
* DUE DATE   : 6/23/21
*****************************************************************************/

#ifndef HEADER_H_
#define HEADER_H_

#include <iostream> // cin, cout.
#include <string>   // string datatype variables.
#include <fstream>  // Fstream files.
#include <iomanip>  // fixed, setw, setprecision.
#include <ostream>  // Ostream data type.
#include <sstream>  // Ostringstream data type.
using namespace std;

/*****************************************************************************
 * Sort Ar(Array)
 *  This function will take in an integer type Array and sort all the numbers
 *  using insertion sort
 *
 *  ==> returns nothing.
 *****************************************************************************/
void SortAr(const int AR_SIZE, // IN & CALC - Array size.
            int Array[]);      // IN & CALC - Integer Array.

/*****************************************************************************
 * SeqSearch (Sequential Search)
 *  This function will take in an integer type Array and make a sequential
 *  search of a user chosen number.
 *
 *  ==> returns approriate index of the array.
 *****************************************************************************/
int SeqSearch(const int AR_SIZE, // IN & CALC - Array size.
              int Array[]);      // IN & CALC - Integer Array.

/*****************************************************************************
 * BiSearch (Binary Search)
 *  This function will take in an integer type Array and make a binary search
 *  of a user chosen number.
```

```
45   *
46   *  ==> returns approriate index of the array.
47                                                                              ↵
   ********************************************************************************/
48  int BiSearch(const int AR_SIZE, // IN & CALC - Array size.
49               int Array[],       // IN & CALC - Integer Array.
50               int key);          // IN & CALC - Key to search for.
51
52  /********************************************************************************
53   * SeqSearch (Sequential Search)
54   *  This function will take in an integer type Array and outputs it.
55   *
56   *  ==> returns nothing.
57                                                                              ↵
   ********************************************************************************/
58  int SeqSearch(const int AR_SIZE, // IN & CALC - Array size.
59                int Array[],       // IN & CALC - Integer Array.
60                int key);          // IN & CALC - Key to search for.
61
62  /********************************************************************************
63   * SeqSearch (Sequential Search)
64   *  This function will take in an integer type Array and outputs it.
65   *
66   *  ==> returns nothing.
67                                                                              ↵
   ********************************************************************************/
68  void OutArray(const int AR_SIZE, // IN & CALC - Array size.
69                int Array[]);      // IN & CALC - Integer Array.
70
71  /********************************************************************************
72   * PrintHeaderFile
73   *   This function will output the header information
74   *
75                                                                              ↵
   ********************************************************************************/
76  void PrintHeaderFile(ostream& output,      // IN  - output datatype.
77                       string asName,        // IN  - assignment name
78                       int asNum,            // IN  - assignment number
79                       string studentName,   // IN  - student's name
80                       string classInfo,     // IN  - class that is being taken
81                       char asType,       // IN  - assignment type
82                       long long studentID); // IN  - student ID
83
84
85  #endif
86
```

```cpp
/*****************************************************************
* AUTHOR     : Andrew Gharios
* STUDENT ID : 1449366
* LAB #5     : Binary Search
* CLASS      : CS1B
* SECTION    : M-TH: 5-7:20p
* DUE DATE   : 6/23/21
*****************************************************************/

#include "Header.h"

/*****************************************************************
* Binary Search
*----------------------------------------------------------------
* This program will output an array out of order, then prompt the user
* to do a sequence search on 4 numbers. It will say if each number is
* found or not and say at which index they were found. Then the program
* will sort the array and output it. After that the user will perform
* a binary search 4 times.
*----------------------------------------------------------------
* INPUT:
*  keyInp   - Key for the number user wants to search.
*
* OUTPUT:
*  intArray - Array of ints.
*  index    - Index where number was found.
*****************************************************************/
int main()
{
    /*************************************************************
    * CONSTANTS
    * ----------------------------------------------------------
    * OUTPUT - USED FOR CLASS HEADING
    * ----------------------------------------------------------
    * PROGRAMMER : Programmer's Name
    * CLASS   : Student's Course
    * SECTION    : Class Days and Times
    * LAB_NUM    : Lab Number (specific to this lab)
    * LAB_NAME   : Title of the Lab
    * ----------------------------------------------------------
    * AR_SIZE    : Size of the array.
    *************************************************************/

    const string AS_NAME = "Binary Search";
    const int AS_NUM = 5;
    const string STUDENT_NAME = "Andrew Gharios";
    const string CLASS_INFO = "M-Th 5-7:20p";
    const char AS_TYPE = 'L';
    const long long STUDENT_ID = 1449366;
```

```
50
51      const int AR_SIZE = 8;
52
53      int intArray[AR_SIZE] = { 4, 1, 7, 12, 7, 13, 9, 21 };
54      int index;   // CALC & OUT - Index where number was found.
55      int i;       // CALC       - Index used in a for loop.
56      int keyInp;  // IN & CALC  - Key input to search for by user.
57
58      PrintHeaderFile(cout, AS_NAME, AS_NUM, STUDENT_NAME, CLASS_INFO,
59                      AS_TYPE, STUDENT_ID);
60
61      OutArray(AR_SIZE, intArray);
62
63      for (i = 0; i < 4; i++)
64      {
65          cout << "Enter an integer to search for: ";
66          cin  >> keyInp;
67          cin.ignore(10000, '\n');
68
69          index = SeqSearch(AR_SIZE, intArray, keyInp);
70
71          if (index != AR_SIZE)
72          {
73              cout << "The integer " << keyInp << " was found in index #" <<
                    index << ".";
74              cout << endl << endl;
75          }
76          else
77          {
78              cout << keyInp << " was not found!\n" << endl;
79          }
80
81      }
82
83      cout << endl << "Performing Insertion Sort!\n";
84      cout << endl;
85      SortAr(AR_SIZE, intArray);
86      OutArray(AR_SIZE, intArray);
87
88      for (i = 0; i < 4; i++)
89      {
90          cout << "Enter an integer to search for: ";
91          cin  >> keyInp;
92          cin.ignore(10000, '\n');
93
94          index = BiSearch(AR_SIZE, intArray, keyInp);
95
96          if (index != AR_SIZE)
97          {
```

```
 98             cout << "The integer " << keyInp << " was found in index #" <<
                  index << ".";
 99             cout << endl << endl;
100         }
101         else
102         {
103             cout << keyInp << " was not found!\n" << endl;
104         }
105
106     }
107
108
109     return 0;
110
111 }
```

```
/*****************************************************************************
* AUTHOR     : Andrew Gharios
* STUDENT ID : 1449366
* LAB #5     : Binary Search
* CLASS      : CS1B
* SECTION    : M-TH: 5-7:20p
* DUE DATE   : 6/23/21
*****************************************************************************/

/*****************************************************************************
 * SeqSearch
 *  This function will search for the inputted key in sequence.
 *
 * INPUTS:
 *  AR_SIZE : Array size.
 *  Array[] : Array to output.
 *  key     : key to search for.
 *
 * OUTPUTS:
 *  index   : Index where the key was found.
 *                                                                           */
int SeqSearch(const int AR_SIZE, // IN & CALC - Array size.
              int Array[],       // IN & CALC - Integer Array.
              int key)           // IN & CALC - Key to search for.
{
    int  index; // CALC - Index for array manipulation.
    bool found; // CALC - Boolean if the search value was found.

    index = 0;
    found = false;

    while (!found && index < AR_SIZE)
    {
        if (Array[index] == key)
        {
            found = true;
        }
        else
        {
            index++;
        }
    }
    return index;

}
```

```
46
47  /*****************************************************************************
48   * BiSearch
49   *  This function will search for the inputted key in binary.
50   *
51   * INPUTS:
52   *  AR_SIZE : Array size.
53   *  Array[] : Array to output.
54   *  key     : key to search for.
55   *
56   * OUTPUTS:
57   *  index   : Index where the key was found.
58                                                                              ↵
   *****************************************************************************/
59  int BiSearch(const int AR_SIZE, // IN & CALC - Array size.
60               int Array[],       // IN & CALC - Integer Array.
61               int key)           // IN & CALC - Key to search for.
62  {
63      int  index;  // CALC - Index to store a value.
64      int  bott;   // CALC - Bottom value.
65      int  top;    // CALC - Top value.
66      int  mid;    // CALC - Middle value.
67      bool found;  // CALC - Boolean if the search value was found.
68
69      bott  = 0;
70      top   = AR_SIZE - 1;
71      found = false;
72
73      while (!found && bott <= top)
74      {
75          mid = (bott + top) / 2;
76
77          if (Array[mid] == key)
78          {
79              found = true;
80              index = mid;
81          }
82          else if (Array[mid] < key)
83          {
84              bott = mid + 1;
85          }
86          else
87          {
88              top = mid - 1;
89          }
90      }
91      if (!found)
92      {
93          index = AR_SIZE;
```

```
94      }
95
96      return index;
97  }
```

```cpp
/*****************************************************************************
* AUTHOR     : Andrew Gharios
* STUDENT ID : 1449366
* LAB #5     : Binary Search
* CLASS      : CS1B
* SECTION    : M-TH: 5-7:20p
* DUE DATE   : 6/23/21
*****************************************************************************/
#include "Header.h"

/*****************************************************************************
 * SeqSearch
 *  This function will sort the inputted array and put it in order.
 *
 * INPUTS:
 *  AR_SIZE : Array size.
 *  Array[] : Array to output.
 *
 * No Outputs.
 *
*****************************************************************************/
void SortAr(const int AR_SIZE, // IN & CALC - Array size.
            int Array[])   // IN & CALC - Integer Array.
{
    int index; // CALC - Index for array manipulation.
    int j;     // CALC - Index named j for array maniplation.
    int temp;  // CALC - Temporary storage for an integer.

    for (index = 1; index < AR_SIZE; index++)
    {
        temp = Array[index];
        j = index - 1;

        while (j >= 0 && Array[j] > temp)
        {
            Array[j + 1] = Array[j];
            j = j - 1;
        }
        Array[j + 1] = temp;
    }
}
```

```cpp
/*****************************************************************************
**
* AUTHOR     : Andrew Gharios
* STUDENT ID : 1449366
* LAB #5     : Binary Search
* CLASS      : CS1B
* SECTION    : M-TH: 5-7:20p
* DUE DATE   : 6/23/21
*****************************************************************************
/
#include "Header.h"

/*****************************************************************************
 * OutArray
 *  This function will output the array that is inputted.
 *
 * INPUTS:
 *  AR_SIZE : Array size.
 *  Array[] : Array to output.
 *
 * No Ouputs.
 *
*****************************************************************************/
void OutArray(const int AR_SIZE, // IN & CALC - Array size.
              int Array[])       // IN & CALC - Integer Array.
{
    int index; // CALC - Index for array manipulation.

    for (index = 0; index < AR_SIZE; index++)
    {
        cout << "Index #" << index << ": " << Array[index] << endl;
    }
    cout << endl;
}
```

```cpp
#include "Header.h"

/*****************************************************************************
 * PrintHeaderFile
 *   This function will output the header information
 *
 * PRE-CONDITIONS
 *   The following parameters need to have a defined value prior to calling
 *   the function
 *         asName: The name of the assignment given in the course
 *         asNum: The number of the assignment given in the course
 *         studentName: The name of the student writing the code
 *         classInfo: The course name, date, and time of the class
 *         asType: Will either output as a lab or an assignment
 *         studentID: The Identification Number of the student
 *
*****************************************************************************/

void PrintHeaderFile(ostream& output,     // IN - output datatype.
                     string asName,       // IN - assignment name
                     int asNum,           // IN - assignment number
                     string studentName,  // IN - student's name
                     string classInfo,    // IN - class that is being taken
                     char asType,         // IN - assignment type
                     long long studentID) // IN - student ID
{
    output << left;
    output << "*************************************************************
      \n";
    output << "*   PROGRAMMED BY : " << studentName << endl;
    output << "*   " << setw(14) << "STUDENT ID " << ": " << studentID << endl;
    output << "*   " << setw(14) << "CLASS " << ": " << classInfo << endl;
    output << "*   ";

    // PROCESSING - This will adjust setws and format appropriately based
    //              on if this is a lab 'L' or assignment

    if (toupper(asType) == 'L')
    {
        output << "LAB #" << setw(9);
    }
    else
    {
        output << "ASSIGNMENT #" << setw(2);
    }
    output << asNum << ": " << asName << endl;
    output << "*************************************************************";
    output << right << endl;
```

```
47
48      return;
49  }
```