Instructor: Dr. Jie Shen
Release date: Month. 01, 2021
Due date: Month. 08, 2021

Student Name: Demetrius Johnson

*Special note: see my lab uploads of the .CPP and .EXE files for ease of access and testing any of the programs for any question.

Table of Contents

Question 1

Source code (USED C++ COMPILER on Microsoft Windows 10)

```
// CIS-200-LAB_6-DemetriusJohnson.cpp : This file contains the 'main' function. Program execution
begins and ends there.
//

/*
//Author: Demetrius E Johnson
//Date: 08 MONTH 2021
//Last Modification Date: 03-08-2021
//Purpose: test the use of a recursive function


*/

/*
Question 1:

Write a recursive function defined by the following recursive formula:

foo (Y, X) =
   Y                                        if  X = 1
   1                                        if  X = Y
   (foo ( Y-1, X-1) + 4* foo ( Y-1, X))     if Y > X  > 1

Write a driver to print out the value for foo (5, 3)    and foo (6, 5).

In addition, print out the total number of recursive function calls in each case.



*/



#include <iostream>
//#include<assert.h>
using namespace std;


//FUNCTION DECLARATIONS

int recursiveFunction(int Y, int X, int& numRecFunc_calls);

//FUNCTION DECLARATIONS



int main()
{
   int numberOfRecFCalls = -1; //first call will cause this to go to 0; the first call is not a recursive call

   cout << "-----WELCOME: This program tests the use of a recursive function ----by Demetrius
Johnson\n\n\n";
```

```
   cout << "Output for recursive function calls for input (5, 3): " << recursiveFunction(5, 3,
numberOfRecFCalls) << endl;
   cout << "Number of recursive calls of the function: " << numberOfRecFCalls;

   numberOfRecFCalls = -1;
   cout << "\n\nOutput for recursive function calls for input (6, 5): " << recursiveFunction(6, 5,
numberOfRecFCalls) << endl;
   cout << "Number of recursive calls of the function: " << numberOfRecFCalls;

   cout << endl << endl << "The program has finished execution....now exiting...thank you....\n\n";
   system("pause");

   return 0;
}


//FUNCTION DEFINITIONS BELOW THIS LINE

int recursiveFunction(int Y, int X, int& numRecFunc_calls) {

   numRecFunc_calls++;
   if (X == 1) { return Y; }
   if (X == Y) { return 1; }
   if(X > 1 && X < Y) { return recursiveFunction((Y - 1), (X - 1), numRecFunc_calls) + (4 *
recursiveFunction(Y - 1, X, numRecFunc_calls)); }

}
```

Test data and expected results

Test Table:

Test #	Valid / Invalid Data	Description of test	Input Value	Expected Output	Actual Output	Test Pass / Fail
1	valid	Test recursive function	(5,3)	Output 67; number of recursive calls: 10	See screenshot	pass
2	valid	Test recursive function	(6,5)	Output 18; number of recursive calls: 8	See screenshot	pass

TEST 1:

```
C:\Users\ferve\OneDrive\Documents\WINTER 2021 SEMESTER CLASS FILES\CIS 200 - RETAKE - WINTER 2021 -JIE SHEN\Lab...

-----WELCOME: This program tests the use of a recursive function ----by Demetrius Johnson


Output for recursive function calls for input (5, 3): 67
Number of recursive calls of the function: 10

Output for recursive function calls for input (6, 5): 18
Number of recursive calls of the function: 8

The program has finished execution....now exiting...thank you....

Press any key to continue . . .
```

TEST 2:

```
C:\Users\ferve\OneDrive\Documents\WINTER 2021 SEMESTER CLASS FILES\CIS 200 - RETAKE - WINTER 2021 -JIE SHEN\Lab...

-----WELCOME: This program tests the use of a recursive function ----by Demetrius Johnson


Output for recursive function calls for input (5, 3): 67
Number of recursive calls of the function: 10

Output for recursive function calls for input (6, 5): 18
Number of recursive calls of the function: 8

The program has finished execution....now exiting...thank you....

Press any key to continue . . .
```

Question 2

Source code (USED C++ COMPILER on Microsoft Windows 10)

```
// CIS-200-LAB_6-DemetriusJohnson.cpp : This file contains the 'main' function. Program execution
begins and ends there.
//

/*
//Author: Demetrius E Johnson
//Date: 08 MONTH 2021
//Last Modification Date: 03-08-2021
//Purpose: implement a recursive function that conducts a binary search


*/

/*
Question 2:

Implement a recursive function that conducts a binary search with the following interface:

bool BinarySearch (int info[ ],  int x,  int fromLoc, int toLoc, int &step);
//  info[fromLoc, toLoc] stores an ordered list
// step:  the number of search step of this binary search
//  x --  a search item
//  if x is in the list, return true; otherwise, return false


Test cases:
#define LEN 10000
int info[LEN];
for(int i=0; i< LEN; i++)
   info[i] = i;

int step;

cout << BinarySearch(info, 997, 0, LEN-1, step)  << endl;
cout << "Binary search steps: " << step << endl;

cout << BinarySearch(info, 20000, 0, LEN-1, step) << endl;
cout << "Binary search steps: " << step << endl;



*/



#include <iostream>
#define LEN 10000
//#include<assert.h>
using namespace std;
```

```
//FUNCTION DECLARATIONS
bool BinarySearch(int info[], int x, int fromLoc, int toLoc, int& step);
//FUNCTION DECLARATIONS



int main()
{

   cout << "-----WELCOME: This program tests the implementation of a recursive function that conducts a
binary search.\n----by Demetrius Johnson\n\n";
   cout << "Note that the Time Complexity for a binary search is log_2(N)\n\n\n";
   int info[LEN];

   for (int i = 0; i < LEN; i++) { info[i] = i; } //set every element of the array equal to its element position (0
- 9999)

   int step = 0; //initiate step counter to 0 steps --> first call to binarySearch recursive function

   cout << "For binary search of 9997 from sorted info array 0-9999 ";
   cout << "(True = 1 or False = 0): " << BinarySearch(info, 9997, 0, LEN - 1, step) << endl << endl;
   cout << "Binary search steps: " << step << endl << endl;
   cout << "Expected binary search steps based on known Time Complexity of log_2(N): log_2(9997) =
13.287\n\n\n\n\n";

   step = 0; //reset step counter
   cout << "For binary search of 20000 from sorted info array 0-9999 ";
   cout << "(True = 1 or False = 0): " << BinarySearch(info, 20000, 0, LEN - 1, step) << endl << endl;
   cout << "Binary search steps: " << step << endl << endl;
   cout << "Expected binary search steps based on known Time Complexity of log_2(N): log_2(10000) =
13.287; \n20000 is not in the array so expect search to be worst case.\n\n";





   cout << endl << endl << "The program has finished execution....now exiting...thank you....\n\n";
   system("pause");

   return 0;
}


//FUNCTION DEFINITIONS BELOW THIS LINE



//  info[fromLoc, toLoc] stores an ordered list
// step:  the number of search step of this binary search
//  x --  a search item
//  if x is in the list, return true; otherwise, return false

bool BinarySearch(int info[], int x, int fromLoc, int toLoc, int& step) {
```

```
    int midPoint = (fromLoc + toLoc) / 2; //set new midpoint every time this function is called using the
passed in from and to location values
    int From_To_Distance = toLoc - fromLoc;
    step++; //increase by the number of times this function is called

    if (x == info[midPoint]) { return true; }

    if (x > info[midPoint] && From_To_Distance > 1) { //ensure from and to location difference is greater
than 1 so that the function will not loop


        fromLoc = midPoint;
        return BinarySearch(info, x, fromLoc, toLoc, step);
    }

    if (x < info[midPoint] && From_To_Distance > 1) { //ensure from and to location difference is greater
than 1 so that the function will not loop

        toLoc = midPoint;
        return BinarySearch(info, x, fromLoc, toLoc, step);
    }

    if (From_To_Distance < 2) { //need this for the case where FROM and TO is a difference of only 1 or 0
in terms of position;
                                //EXAMPLE: (1+2)/2  = 1 --> (1+2)/2 = 1....repeating from above functions assigning
from and to and a new midpoint that will keep resulting in 1; same scenario for FROM = TO.

        if (x == info[fromLoc] || x == info[toLoc]) { return true; }
        else { return false; }

    } //value not found case (or valued found at the last two locations to search)
}
```

Test data and expected results

Test Table:

Test #	Valid / Invalid Data	Description of test	Input Value	Expected Output	Actual Output	Test Pass / Fail
1	valid	Test binary recursive search function	9997	True (1),steps = 13	See screenshot	pass
2	valid	Test binary recursive search function	20000	False (0),steps = 15	See screenshot	pass

TEST 1:

```
C:\Users\ferve\OneDrive\Documents\WINTER 2021 SEMESTER CLASS FILES\CIS 200 - RETAKE - WINTER 2021 -JIE SHEN\Labs\Lab 6\Execut...
-----WELCOME: This program tests the implementation of a recursive function that conducts a binary search.
----by Demetrius Johnson

Note that the Time Complexity for a binary search is log_2(N)


For binary search of 9997 from sorted info array 0-9999 (True = 1 or False = 0): 1

Binary search steps: 13

Expected binary search steps based on known Time Complexity of log_2(N): log_2(9997) = 13.287



For binary search of 20000 from sorted info array 0-9999 (True = 1 or False = 0): 0

Binary search steps: 15

Expected binary search steps based on known Time Complexity of log_2(N): log_2(10000) = 13.287;
20000 is not in the array so expect search to be worst case.



The program has finished execution....now exiting...thank you....

Press any key to continue . . .
```

TEST 2:

```
C:\Users\ferve\OneDrive\Documents\WINTER 2021 SEMESTER CLASS FILES\CIS 200 - RETAKE - WINTER 2021 -JIE SHEN\Labs\Lab 6\Execut...
-----WELCOME: This program tests the implementation of a recursive function that conducts a binary search.
----by Demetrius Johnson

Note that the Time Complexity for a binary search is log_2(N)


For binary search of 9997 from sorted info array 0-9999 (True = 1 or False = 0): 1

Binary search steps: 13

Expected binary search steps based on known Time Complexity of log_2(N): log_2(9997) = 13.287



For binary search of 20000 from sorted info array 0-9999 (True = 1 or False = 0): 0

Binary search steps: 15

Expected binary search steps based on known Time Complexity of log_2(N): log_2(10000) = 13.287;
20000 is not in the array so expect search to be worst case.



The program has finished execution....now exiting...thank you....

Press any key to continue . . .
```

Question 3

Source code (USED C++ COMPILER on Microsoft Windows 10)

```
// CIS-200-LAB_6-DemetriusJohnson.cpp : This file contains the 'main' function. Program execution
begins and ends there.
//

/*
//Author: Demetrius E Johnson
//Date: 08 MONTH 2021
//Last Modification Date: 03-08-2021
//Purpose: Show use of a Linear search


*/

/*
Question 3:

Implement a function that conducts a linear search with the following interface:

bool LinearSearch (int info[ ],  int x,  int fromLoc, int toLoc, int &step);
//  info[fromLoc, toLoc] stores an ordered list
// step:  the number of search step of this linear search
//  x --  a search item
//  if x is in the list, return true; otherwise, return false


Test cases:
#define LEN 10000
int info[LEN];
for(int i=0; i< LEN; i++)
   info[i] = i;

int step;

cout << LinearSearch(info, 997, 0, LEN-1, step)  << endl;
cout << "Linear search steps: " << step << endl;

cout << LinearSearch(info, 20000, 0, LEN-1, step) << endl;
cout << "Linear search steps: " << step << endl;



*/



#include <iostream>
#define LEN 10000
//#include<assert.h>
using namespace std;
```

```
//FUNCTION DECLARATIONS
bool LinearSearch(int info[], int x, int fromLoc, int toLoc, int& step);
//FUNCTION DECLARATIONS



int main()
{

   cout << "-----WELCOME: This program tests the implementation of a linear function that conducts a
linear search.\n----by Demetrius Johnson\n\n";
   cout << "Note that the Time Complexity for a linear search O(N)\n\n\n";
   int info[LEN];

   for (int i = 0; i < LEN; i++) { info[i] = i; } //set every element of the array equal to its element position (0
- 9999)

   int step = 0; //initiate step counter to 0 steps --> first call to linearSearch function

   cout << "For linear search of 9997 from sorted info array 0-9999 ";
   cout << "(True = 1 or False = 0): " << LinearSearch(info, 9997, 0, LEN - 1, step) << endl << endl;
   cout << "Linear search steps: " << step << endl << endl;
   cout << "Expected linear search steps based on known Time Complexity of O(N): 9998 (0 to 9997 -->
9998 steps)\n\n\n\n\n";

   step = 0; //reset step counter
   cout << "For linear search of 20000 from sorted info array 0-9999 ";
   cout << "True = 1 or False = 0: " << LinearSearch(info, 20000, 0, LEN - 1, step) << endl << endl;
   cout << "Linear search steps: " << step << endl << endl;
   cout << "Expected linear search steps based on known Time Complexity of O(N): 10000. \n20000 is not
in the array so expect search to be worst case.\n\n";





   cout << endl << endl << "The program has finished execution....now exiting...thank you....\n\n";
   system("pause");

   return 0;
}


//FUNCTION DEFINITIONS BELOW THIS LINE



//  info[fromLoc, toLoc] stores an ordered list
// step:  the number of search step of this linear search
//  x --  a search item
//  if x is in the list, return true; otherwise, return false

bool LinearSearch(int info[], int x, int fromLoc, int toLoc, int& step) {
```

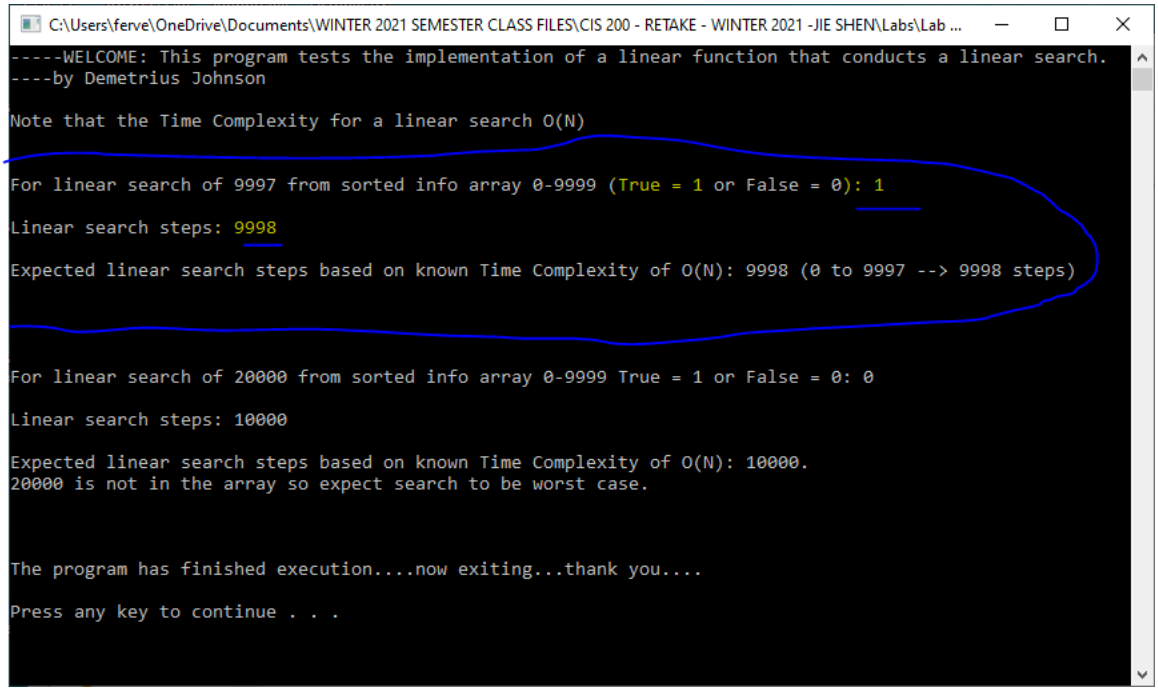
```
    for(int i = fromLoc; i <= toLoc; i++){

        step++;
        if (x == info[i]) { return true; }
        if (i == toLoc) { return false; } //if i increments all the way to toLoc without finding a match from the statement above this, then value is not found so return false

    }



}
```

Test data and expected results

Test Table:

Test #	Valid / Invalid Data	Description of test	Input Value	Expected Output	Actual Output	Test Pass / Fail
1	valid	Test linear search function	9997	True (1),steps = 9998	See screenshot	pass
2	valid	Test linear search function	20000	False (0),steps = 10000	See screenshot	pass

TEST 1:



TEST 2:

```
C:\Users\ferve\OneDrive\Documents\WINTER 2021 SEMESTER CLASS FILES\CIS 200 - RETAKE - WINTER 2021 -JIE SHEN\Labs\Lab ...

-----WELCOME: This program tests the implementation of a linear function that conducts a linear search.
----by Demetrius Johnson

Note that the Time Complexity for a linear search O(N)


For linear search of 9997 from sorted info array 0-9999 (True = 1 or False = 0): 1

Linear search steps: 9998

Expected linear search steps based on known Time Complexity of O(N): 9998 (0 to 9997 --> 9998 steps)



For linear search of 20000 from sorted info array 0-9999 True = 1 or False = 0: 0

Linear search steps: 10000

Expected linear search steps based on known Time Complexity of O(N): 10000.
20000 is not in the array so expect search to be worst case.


The program has finished execution....now exiting...thank you....

Press any key to continue . . .
```

Submission

Provide an MS word document or a pdf file with the following information:

1. Cover page with lab number and title, your name, date
2. Test data and expected results
3. Running log/output (screen shots should be provided)
4. Insert the source code and screenshots into the word document
5. You should submit your work through Canvas site for cis 200 lab.
6. The filename of the word document should follow the convention: FirstName_LastName_Cis200Lab3.doc or pdf.