CIS 200 - Lab 0601

1. Problem Statement

Create a recursive function that performs a linear search within an array. Eventually, scale up the array to cause a stack overflow.

2. Requirements

2.1 Assumptions

- User will only input integers
- User will only input into the command line

2.2 Specifications

- User will input key to search for
- Program will alert when found
- Program will alert how many function calls existed

3. Decomposition Diagram

- Program
    - Input
        - Take in User Key
    - Process
        - Recursively Compare Key to Data
    - Output
        - Data Found after X calls
        - Not Found

4. Test Strategy

- Valid Data
- Invalid Data

5. Test Plan Version 1

Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	Key within bounds				

Invalid Data	1	Key out of bounds				
Invalid Data	2	Key out of bounds				
Invalid Data	3	Cause Overflow				

6. Initial Algorithm

1. Create Function
    a. Name: recursiveLinearSearch
    b. Parameters: integer array, integer key, integer size, bool methodStatus
    c. Return: recursiveLinearSearch
    d. Method
        i. Decrement Size for next use
        ii. If key larger than size
            1. Return -1
        iii. Else If key matches array index @ size
            1. Set methodStatus to true
            2. Return Global SIZE minus current size/function call
        iv. Else
            1. Return calling recursiveLinearSearch
2. Create Main
    a. Declare Global SIZE as 5000
    b. Declare bool methodStatus as false
    c. Declare integer recusiveCalss as 0
    d. Declare integer key as 0
    e. Declare integer Array with size of SIZE
    f. Fill Array with 0 to SIZE - 1
    g. Prompt user for new key
    h. Perform recursiveLinearSearch with key
    i. If methodStatus after call is true
        i. Alert Key found after number of recursiveCalls
    j. Else
        i. Alert Key was not found.

7. Test Plan Version 2

Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	Key within bounds	1234	Found after 3766 calls		
Invalid Data	1	Key out of bounds	-1	Key not found		
Invalid Data	2	Key out of bounds	5001	Key not found		
Invalid Data	3	Cause Overflow	Array SIZE	Stack Overflow		

8. Code

```
//Program Name: Recursive Linear Search
//Programmer Name: Arthur Aigeltinger IV
//Description: Use recursion to create a linear searching function.
//Date Created: 10/23/18

#include <iostream>

//Global Variables
const int SIZE = 5000;

//Function Prototypes
int recursiveLinearSearch(int array[], int key, int size, bool &methodStatus);

int main()
{
	//Declare Variables
	bool methodStatus = false;
	int recursiveCalls = 0;
	int key = 0;

	//Declare Array
	int intArray[SIZE];

	//Fill Array
	for (int i = 0; i < SIZE; i++)
	{
		intArray[i] = i;
	}

	std::cout << "Please enter a 'key' or number to search for: ";
```

```
	std::cin >> key;

	recursiveCalls = recursiveLinearSearch(intArray, key, SIZE, methodStatus);

	if (methodStatus)
	{
		std::cout << "Found Key " << key << " after " << recursiveCalls << "
recursive calls" << std::endl;
	}
	else
	{
		std::cout << "Key was not found" << std::endl;
	}

	system("pause");
	return 0;
}

//Description: Recursively performs a liner search within a given sorted array.
//Pre-Condition: Array being filled
//Post-Condition: User will be satisfied.
int recursiveLinearSearch(int array[], int key, int size, bool & methodStatus)
{
	size--;	//Decrement Size

	if (size < 0 || key < 0 || key > size) //Checking for valid size and key from
the start
	{
		return -1;
	}
	else if (array[size] == key)			//When key is found
	{
		methodStatus = true;
		return SIZE - size;
	}
	else							//Recursive Call
	{
		return recursiveLinearSearch(array, key, size, methodStatus);
	}

	return 0;
}
```

9. Updated Algorithm

1. Create Function
    a. Name: recursiveLinearSearch

b. Parameters: integer array, integer key, integer size, bool methodStatus
c. Return: recursiveLinearSearch
d. Method
    i. Decrement Size for next use
    ii. ~~If key larger than size~~
    iii. If key is not within bounds
        1. Return -1
    iv. Else If key matches array index @ size
        1. Set methodStatus to true
        2. Return Global SIZE minus current size/function call
    v. Else
        1. Return calling recursiveLinearSearch

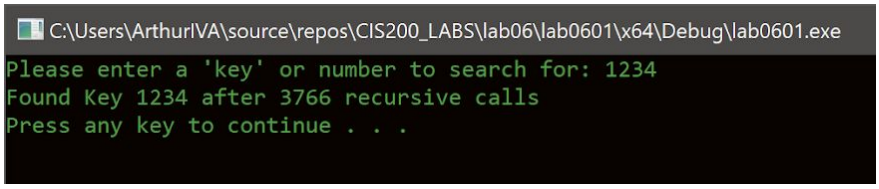
2. Create Main
    a. Declare Global SIZE as 5000
    b. Declare bool methodStatus as false
    c. Declare integer recusiveCalss as 0
    d. Declare integer key as 0
    e. Declare integer Array with size of SIZE
    f. Fill Array with 0 to SIZE - 1
    g. Prompt user for new key
    h. Perform recursiveLinearSearch with key
    i. If methodStatus after call is true
        i. Alert Key found after number of recursiveCalls
    j. Else
        i. Alert Key was not found.

10. Test Plan Version 3

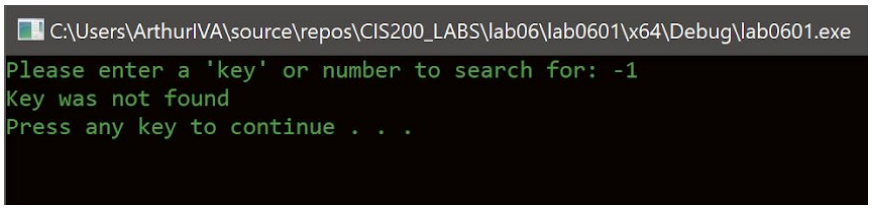
Test Strategy	#	Description	Input	Expected Output	Actual Output	Pass/Fail
Valid Data	1	Key within bounds	1234	Found after 3766 calls	Found after 3766 calls	Pass
Invalid Data	1	Key out of bounds	-1	Key not found	Key not found	Pass
Invalid Data	2	Key out of bounds	5001	Key not found	Key not found	Pass
Invalid Data	3	Cause Overflow	Array SIZE 50000	Stack Overflow	Stack Overflow	Pass

11. Screenshots

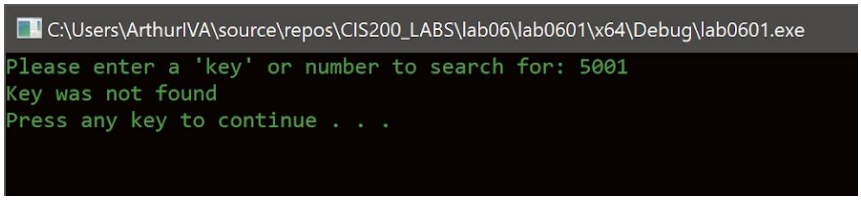
Valid Test Case 1



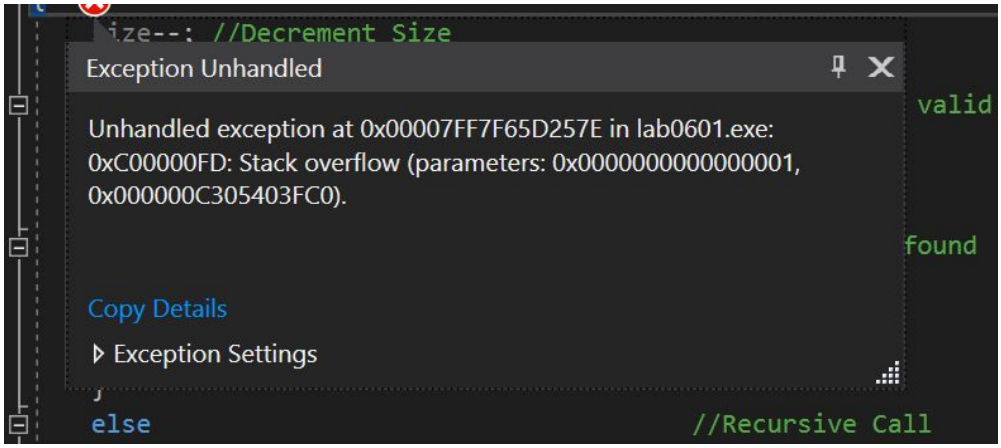
Invalid Test Case 1



Invalid Test Case 2



Invalid Test Case 3



12. Error Log

Error Type (Logic/Runtime)	Cause of Error	Solution to Error
Logic/Runtime	Checking Key compared to size and not within bounds.	Including checking for within bounds. I'm not entirely sure why this caused an error as

		logically it shouldn't, but this solution offers some redundancy.

13. Status

Program WORKS!