**Submission of Your Work**

You need to prepare and submit ONE SINGLE MS Word document to Canvas (in your lab section) as LastName\_FirstName\_Labxy.doc. It must contain:

* Your NAME only on page 1
* For ***each*** question:
  + Specify the question number.
  + Source code. Copy/Paste your final source code. You must include standard “comment header” even if code is provided. *Do Not* paste a snippet of your source code, it must be copy/pasted.
  + Initial test plan. After reading the question requirements, but **before** beginning any coding, create the test case table, below, completed through column Expected Output. Include in your report.
  + Final test plan. Write your program then complete the **test table** with actual output results and include in your report *AFTER* your source code.
  + Output results. Paste in a snippet of output showing results for **every listed test case in your final test plan**, labeled with test case #

Test Table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |

* Add / delete rows from Test Table as necessary
* Modify column widths as necessary
* Test both valid and invalid input
* Test for every output expected
* If failure is an expected output and it happens then that test Passes
* Any test that fails means the program must be fixed so that it passes the test
  + Failing tests need a new test row, ie 1a, 1b, etc, showing corrections from original

**Program 1:**

You are to create a recursive function to perform a linear search through an array.

**How Program Works**

* Program has array size of 5000
* Load values into the array, equal to its index value. Index 5 has value 5, index 123 has value 123.
* Pass**array, key, and size to the recursive function:** 
  + **int recursiveLinearSearch(int array[],int key, const int size, bool & methodStatus)**
* User enters key to search for, recursive function calls itself until the key is found (methodStatus is true), print out the key and number of function calls when control is returned to the main
  + Handle situation of key not found (return number of function calls AND methodStatus of false) – print not found message and number of calls in the main
* Function returns a count of how many recursive calls were made
  + Value returned is the number of calls made to that point, that is, when item is found the value returned is 1 with preceding functions adding 1 to the return value until the actual number of recursive function calls are counted).

Determine smallest key value that causes stack-overflow to occur, even if you need to make array larger than 5000.

Test cases need to include (not limited to) biggest possible key value, “not found” message, and a stack overflow condition.

**Program 2:**

You are to create a recursive function to perform a factorial calculation.

* **int factorial(const int value)**
* return -1 if any negative number passed into the function
* Calculate the factorial of the number entered by the user

Determine value at which stack overflow occurs.