Test 01

CSci 515 Spring 2015

<2015-02-24 Tue>

Dates:

Due: In Lab, Wednesday March 11, by 4:05 pm (lab end time)

Description

You have 2 hours (our regularly scheduled lab time) to complete the following tasks. Create a single file, named test01.cpp, in a visual studio project called Test01. Set up the Visual Studio projects using the normal settings for our class and labs, as we have practiced for the past 7 weeks.

Perform the following tasks:

1. In the main function of your program, write a statement that prompts the user for a seed value. Seed the C standard library random number generator with this seed value using the `srand` standard library function. Write an index controlled loop that uses an index to execute 10 times. Inside of your loop, generate a random number between 1 and 4, and display the random number to standard output. If the number is a 1, display a special message "I generated a 1!". The output from this task and loop in your main function should look like the first part of the example output given below.

2. After the previous task in your main function, write code to perform the following task. I have given you a file named "test-01-data.txt". This file has a single integer value on each line. Open this file, and inside of a sentinel controlled loop, read in the values from the file and display the values to standard output. Your sentinel controlled loops should simply be a while loop that executes until the read from the file fails. The output from reading the give file will be exactly as given in the example output shown below.

3. Write a function named `calculateHypotenuse`. The function will take two floating point values as parameters, named `a` and `b`. The function should calculate the hypotenuse of a right triangle using the `a` and `b` parameters, according to this formula:

$$c = \sqrt{a^2 + b^2}$$

e.g. the hypotenuse is the square root of adding together the square of a and b. This value should be calculated and returned (as a floating point result) as a result of calling your function. In your main function, show an example of calling your function with values of 3.0 and 4.0 for the `a` and `b` parameters.

4. Create an array of integers of size 10 named values. Create a function named `initNegative` that takes an array of integers and a size as its two input parameters. This will be a void function (it will return no result). This functions should initialize each value to be equal to the negative of the index, e.g. the value at index 0 will be 0, the value at index 1 will be -1, the value at index 2 will be -2, etc. Call your function in `main()` to initialize the array. In `main`, after initializing the array, write a loop that displays the contents of the array. An example of the output from this task is shown below as the last part of the example output.

Your program output for the 4 previous tasks should look something close to the following when I run your program:

```
----- Task 1 -----
Enter a seed value: 42
0: 3
1: 1
I generated a 1!
2: 2
3: 2
4: 1
I generated a 1!
5: 3
6: 2
7: 1
I generated a 1!
8: 4
```

```
9: 4

----- Task 2 -----
Read value from file: 3
Read value from file: 1
Read value from file: 42
Read value from file: 9
Read value from file: 11
Read value from file: 12
Read value from file: 7

----- Task 3 -----
Hypotenuse of triangle with sides 3 and 4: 5

----- Task 4 -----
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
```

Test Submission

An eCollege dropbox has been created for this test. You should upload your version of the test by the end of test time to the eCollege dropbox named `Test 01`. Work submitted by the end of the allotted time will be considered, but after the test ends you may no longer submit work, so make sure you submit your best effort by the test end time in order to receive credit.

Requirements and Grading Rubrics

Program Execution, Output and Functional Requirements

1. Your program must compile, run and produce some sort of output to be graded. You will loose at least 1/3 of the total points (33) if your program does not compile and run when submitted.

2. 10 pts (1 letter grade). Up to 1 letter grade will be awarded for formatting and style issues for the test. Your program must meet (most) all of the standard class style/formatting guidelines that we have been practicing and using in our labs and assignments for this course.

3. 20 pts. Task 1. You must use an index controlled for loop, and have an if statement. Your output for this task must be as shown in the example output.

4. 20 pts. Task 2. You must successfully open up and read from the given file. You should use defensive programming to detect when the file is not opened or found correctly. You must use a sentinel controlled loop to read all values from the file. Your output for task 2 must look like that shown in the example output.

5. 25 pts. Task 3. You must correctly name and define the function as required for the task. The function must accept the correct parameters as input, and return the correct result type. The function must be implemented correctly to perform the desired calculation.

6. 25 pts. Task 4. You must correctly name and define the function as required for this task. The function should take the array and the array size as parameters and initialize the array as specified. You must define the array in your main loop and invoke the function with your array to be initialized. You should use a defined constant to specify the size of the array in main. Your should display your array after being initialized, as shown in the example output.

Program Style

Your programs must conform to the style and formatting guidelines given for this course. The following is a list of the guidelines that are required for the lab to be submitted this week.

1. The file header for the file with your name and program information and the function header for your main function must be present, and filled out correctly.

2. A function header must be present for all functions you define. You must document the purpose, input parameters and return values of all functions. Your function headers must be formatted exactly as shown in the style guidelines for the class.

3. You must indent your code correctly and have no embedded tabs in your source code. (Don't forget about the Visual Studio Format Selection command).

4. You must not have any statements that are hacks in order to keep your terminal from closing when your program exits (e.g. no calls to system()).

5. You must have a single space before and after each binary operator.

6. You must have a single blank line after the end of your declaration of variables at the top of a function, before the first code statement.

7. You must have a single blank space after , and ; operators used as a separator in lists of variables, parameters or other control structures.

8. You must have opening `{` and closing `}` for control statement blocks on their own line, indented correctly for the level of the control statement block.

9. All control statement blocks (if, for, while, etc.) must have `{ }` enclosing them, even when they are not strictly necessary (when there is only 1 statement in the block).

   (a) You should attempt to use meaningful variable and function names in your program, for program clarity. Of course, when required, you must name functions, parameters and variables as specified in the assignments. Variable and function names must conform to correct `camelCaseNameingConvention` .

Failure to conform to any of these formatting and programming practice guidelines for this test will result in loosing 1 letter grade You can get a B for this test if you do it perfectly, but have bad or missing style/formatting. To get an A, however, you need to follow (most) of the style/formatting requirements for this course on your test code.