CS 4380 Project 0: Getting Started

NOTE: READ THIS DOCUMENT IN ITS ENTIRETY BEFORE YOU BEGIN WORKING

Introduction:

The purpose of this project is to ensure that you understand and can utilize the expectations and tools that will be required of you as you complete programming projects throughout the semester. Specifically, you will learn and/or practice using the following tools/resources (with reference websites listed):

- Git https://git-scm.com
- GitHub https://github.com (documentation found here: https://docs.github.com/en/get-started/quickstart)
- g++ https://gcc.gnu.org
- gtest (GoogleTest) https://google.github.io/googletest/
- make https://www.gnu.org/software/make/
- cmake https://cmake.org/cmake/help/latest/guide/tutorial/index.html

Part 1

For this project you must develop a simple base conversion program; but you must build this program in stages. For Part 1 you must implement a simple terminal-based base 10 to base 2 conversion tool that meets the following requirements:

Req1) The program must be written in C++.

Req2) The program must correctly perform base 10 to base 2 conversion for integers between 0 and 4294967295 (inclusive)

Req3) Upon execution the program must prompt the user to “Enter a non-negative base 10 integer between 0 and 4294967295 (with no commas) and hit Enter/Return: ”

Req4) The program shall perform input validation to ensure that only valid and properly formatted non-negative integers are entered. If invalid input is entered the program should print the message “Invalid input!” followed by a newline, and then exit by returning 1.

Req5) When valid and properly formatted non-negative base 10 integers are input to the program, the program shall respond by generating and printing to the screen the corresponding base 2 value (as a string of 1s and 0s - with no leading zeros), followed by a newline. The program shall then exit by returning 0.

Req6) The program shall consist of a single c++ file called main.cpp

Req7) The project shall contain a README file that shall contain at a minimum a description of how the program was validated up to this point (i.e. unit tests, manual testing, etc.)

In addition to meeting the program requirements listed above you must also utilize the version control tools git and github in the development of your project. For Part 1 you need to create a new private github repository called CS4380-Project0. Develop Part 1 on a branch called base-ten-to-base-two. Make sure to push your branch to the remote repository on GitHub.

Note: When you have completed Part 1, merge the branch base-ten-to-base-two into main (but DO NOT delete the branch!).

Part 2

For part 2 of this project, the following additional requirements must be met.

Req8) The program must contain a function called convTen2Two(). This function does the work of performing base conversion as described in Req2.

Req9) The function convTen2Two() shall accept a single unsigned int as its lone parameter. The parameter contains the base 10 value to be converted to base 2. This function shall return a std::string that contains the bitwise (characters 0 and 1) representation of the unsigned int (with no leading zeros). The function shall have the following prototype: `std::string convTen2Two(unsigned int baseTenValue);`

Note: If the final computed value is 0, this shall not be considered a leading zero. In other words, if the input parameter is 0, the return value should be the string "0";

Req10) **This requirement supersedes Req6**. The function definition for convTen2Two() shall reside in a file called myUtils.cpp, and its prototype shall reside in a file called myUtils.h. Appropriate header guards etc. shall be used in the .h file. The source file main.cpp shall be modified to accommodate these changes so that the program continues to meet requirements Req1 – Req5.

Req11) The project shall include a Makefile that includes at a minimum rules for:

1) Building an executable called b10tob2 which incorporates main.cpp and myUtils.cpp and meets Req1 – Req5. The rule shall be called “application”.

2) Removing all build artifacts, including object files and executables. This rule shall be called “clean”.

In addition to meeting the program/project requirements listed for Part 2, you must continue to utilize Git/GitHub for version control. You must perform the work for Part 2 in a separate branch (split from the main branch) called modularization. When you are finished with your work (and have committed and pushed it) on Part 2, merge the branch modularization into the main branch (but DO NOT delete the branch).

Part 3

Note: You may find the tutorials at: https://google.github.io/googletest/primer.html and https://www.geeksforgeeks.org/gtest-framework/ useful for completing this portion of the project.

In Part 3 you must incorporate the GoogleTest framework into your project, utilize it to implement at least 2 unit tests for the convTen2Two(), and leverage cmake to automate the build process. You must meet the following requirements:

> Req12) The project shall include a source file called myTests.cpp which shall include at a MINIMUM two test functions that follow the GoogleTest framework's requirements, and which perform validation of Req2.
>
> Req 13) **This requirement supercedes Req11**. The project shall include a CMakeLists.txt file containing commands to create both the executable b10tob2 (as described in Req11) as well as an executable called runTests which utilizes the tests in myTests.cpp to validate the convTen2Two() function in myUtils.cpp. runTests should be linked against gtest_main. Executing the command "cmake ." shall result in a properly configured Makefile such that executing "make" shall result in the creation of both of the executables listed in this requirement.
>
> Req14) The README file specified in Req7 must be updated to reflect Req13

In addition to meeting the program/project requirements listed for Part 3, you must continue to utilize Git/GitHub for version control. You must perform the work for Part 3 in a separate branch (split from the main branch) called cmake-and-testing. When you are finished with your work (and have committed and pushed it) on Part 3, merge the branch cmake-and-testing into the main branch (but DO NOT delete the branch).

What to Turn In:

When you are completed with the project you will need to submit the URL for your repository on the assignment page in Canvas. **Your repository should be private. Instructions for granting access to your repo for the course grader will be provided by your instructor.**

Note: If you are invoking the Non-Late policy please insert a note at the top of your README file that states "INVOKING THE NON_LATE POLICY" followed immediately by your work log. (The rest of you README should still contain any project specified requirements).