Assignment #11

(*Marked Lab – 10 points*)

Problem Solving and Programming in C++
Department of Computer Science
Old Dominion University

White-Box Testing – Lab Session

White-box testing is a type of testing where the tester has full access to the running code unlike black-box testing. It is also known as structural testing, clear box testing, and glass box testing. In white-box testing we will try to choose "smart" tests based on the structure of the code, with minimal reference to the requirements.

A good test will run as many lines of code as possible, testing out the outcome of every one of them. If you are writing in C++ on a system with GCC (GNU Compiler Collection) installed, you can use some built-in UNIX tools to get a detailed report of what is happening when the code runs. This is called code coverage. Code coverage tells you (*the tester*) which parts of the code from the running program has correctly executed. The code that wasn't covered might have bugs and other potential problems. Testers will usually like to know which parts of the code was not covered during these tests.

The `Gcov` is a useful tool that creates a report of what lines were executed and which ones were skipped. However, for codes with a large number of lines, reading the report from the `gcov` could be difficult. The `lcov` is another available free tool for the White Box Testing, and it creates the same information in an HTML form.

Lab Exercise:

In this assignment, you are given the C++ source code file for a simple game program called "Guess". The program asks you to predict the name of a sport based on the number of letters. You will have 10 chances to guess one letter at a time. The game is fun, but playing the game is not the actual task.

Your task for this lab would be to apply white box testing method, which you have learned in class on this program. You have to use `gcov` for that. You will also have to do the same testing using `lcov` which is a graphical and more understandable code coverage tool.

You have to create test result files for each test and submit at the end of the lab session.

Method to use `gcov` and `lcov` tools:

First of all, create a directory in your Z drive and name it Assg11_cslogin where the cslogin is your login ID for the computers at the Department of Computer Science at ODU. Download guess.cpp from BlackBoard to this folder.

As these tools are linux tools you have to use your ODU Linux accounts for this lab exercise. *Please note that you can use the gcov code coverage tool with Code::Blocks, but we are not including that in this lab.*

1. Compile your code using two special arguments as below

   ```
   g++ -fprofile-arcs -ftest-coverage -g filename.cpp -o executablename
   ```

   We can see a binary file and a `gcov` file with extension "`.gcno`".

2. Now run the program with your test case (`./executablename`) it will generate a new gcov file with extension "`.gcda`".

3. Now run the `gcov` tool, which analyzes these "`.gcno`" and "`.gcda`" files and create a code coverage information based on the last execution of the program.

- In order to run gcov, use the command "`gcov guess.cpp`"

4. The execution of `gcov` tool will give you the percentage of the code coverage from the last execution of the program, also it generates "`.gcov`" file which contains line by line coverage information.

5. Open the "`.gcov`" file in your favorite editor to check the code coverage information.

   Important symbols:

Number:	→ Indicates the number of times the line executed.
– :	→ Indicates code was not generated for that line.
####:	→ Indicates code was generated for that line but it was never executed.

   Now we will generate some similar code coverage information for your test cases but using a different tool "`lcov`". Which will generate a graphical and easy to understand information for you.

6. Now run `lcov` with some special arguments as shown below.

   ```
   lcov --capture --directory . --output-file filename.info
   ```

   This will generate an output file as indicated above.

7. Now generate graphical information in "**.html**" format using the following command

   ```
   genhtml filename.info --output-directory .
   ```

   This will generate a "**.html**" file that you can open in any web browser.

   This file will show you the code coverage information in nice graphical form. You can access this file via navigating to your working directory on the lab computer or on the virtual lab.

Submission notes:

- Zip the folder you created and name it as "Assg11_cslogin", where the cslogin is your login ID for the computers at the Department of Computer Science at ODU.
- Submit the .zip file in the respective Blackboard link.