# CS515 Assignment 0

The purpose of this assignment is to help you get familiar with our programming environment and submission system. It will also help you gain practice with basic C++ programming. This assignment will be graded but will not count as heavily as other assignments (40 points instead of 100 points).

Download all starter files from `~cs515/public/0P` on agate. To do this, log into your account on agate, change to your working directory, and then copy the files over to your current directory with the following command:

**`cp ~cs515/public/0P/* .`** ← *don't forget the "dot"*

If you don't yet have an account on agate, please email CS support staff immediately at support@cs.unh.edu and CC me so that I am aware of the situation.

The starter code includes a **`hello.cpp`** program that prompts the user to input the name from `stdin` (i.e. keyboard), and print out a greeting message to `stdout` (i.e. screen).

```
$ ./hello
What is your name?
Wicket W. Warrick            ← user input
Hello World!
Welcome to CS515, Wicket!
```

Currently, if the input contains space characters, e.g. white spaces and tabs, only the first string is read, not the complete line of input. Your job is to fix this problem so that the program will read a line of input, which may include several space-delimited strings, and then display the complete line in response. You should use `getline` (http://www.cplusplus.com/reference/string/string/getline/) to accomplish this. In addition, the program processes one line of name only; blank lines are ignored. Spaces in the string do not need to be trimmed. Here's an example of what the program *should* do…

Sample run 1:

```
$ ./hello
What is your name?
Wicket W. Warrick
Hello World!
Welcome to CS515, Wicket W. Warrick!
```

Sample run 2:

```
What is your name?
                                ← user input a blank line
What is your name?
   Spacious Sam :)
Hello World!
Welcome to CS515,    Spacious Sam :)!
```

Review Lab 0 for instructions on compiling and running C++ programs. Next, follow the instructions below to learn how to use a shell script to test your programs.

## How to test run your program with a shell script:

Once you have the executable, you can run it by typing its name directly at the shell prompt:

```
$ ./hello
What is your name?
Wicket W. Warrick          ← user input
Hello World!
Welcome to CS515, Wicket W. Warrick!
```

Or, you can write a shell script to run multiple test cases. In each test, user inputs are saved into a text file and redirected to the program. Thus, all test runs can be executed using a single shell command. The starter code includes a sample shell script named `test.run` and a text file named **`input0`** that contains the user keyboard input.

Here is a sample run by running the script as a shell command:

```
$ ./test.run
+++++++TEST 0++++++++
What is your name?
Hello World!
Welcome to CS515, Wicket W. Warrick!
```

You need to add more test cases (called input1, input2, and input3, without any file extension) by using more test input files and modifying the `test.run` script. By opening up the file in the text editor of your choice, you can see that the internals of the script are fairly simple:

```
#!/bin/bash -f
echo "+++++++TEST 0++++++++"
./hello < input0

#MORE TESTS TO RUN
#echo "+++++++TEST 1++++++++"
#./hello < input1

#echo "+++++++TEST 2++++++++"
#./hello < input2

#echo "+++++++TEST 3++++++++"
#./hello < input3
```

"`echo`" is a command that is used to write text to the console. Use this command to keep your output readable and your script maintainable. "`./hello < input0`" is the command that you would typically use to run your program and takes the text from the file `input0` and treat it as keyboard input. Everything with a "#" in front of it is a comment. As you can see, adding multiple test files is as easy as adding another line to run your program using a different input file. Now you can run several tests at once just by running `./test.run`.

## How to submit your program:

- Submit the source file named **`hello.cpp`**, the **`makefile`**, and also **`test.run`**
- You should also fill out the **README** file and submit it.
- The name of the files must be exactly as specified. Submit the following files to the appropriate assignment on Mimir:

```
hello.cpp makefile test.run README
```

- Programs that produce compile time errors or warnings will receive a zero mark (even if it might work perfectly on your home computer.) To check this, use the make command with the provided makefile and check the results of the automatic tests on Mimir.
- Make sure to provide proper comments in your source file. You must also include the information below:

```
/** CS515 Assignment 0
File: XXX.cpp
Name: XXX
Section: X
Date: XXX
Collaboration Declaration:
          assistance received from TA, PAC and online resources etc.
          …
*/
```

## Some notes on grading:

- Programs are graded for correctness (output results and code details), following directions and using specified features, documentation and style.
- **To successfully pass the provided sample tests is not an indication of a potential good grade; to fail one or more of these tests is an indication of a potential bad grade.**
- You must test thoroughly your program with your own test data/cases to ensure all the requirements are fulfilled. We will use additional test data/cases other than the sample tests to grade your program.
- Here is a tentative grading scheme for this assignment:

| | |
|---|---|
| Program displays complete input line | 20 |
| Continues to prompt on blank input line | 10 |
| test.run tests at least 3 different input files | 10 |
| **Total** | **40** |