CS 010 - Introduction to Computer Science I

Lab 2 - Hello, You!

**Suggested pre-lab work (you should have completed at least some of these items)**

Zyante sections 2.1 to 2.5, and corresponding Codelab exercises

Video tutorials: [Module 2 playlist](http://www.youtube.com/playlist?list=PLTTJbxrH72A0_7gF5fz_6kfuOjfJiZ48R)

**Collaboration policy**

Collaboration on Piazza, Cloud9 or in person on these lab exercises is strongly ENCOURAGED. They are intended for practice, not assessment -- feel free to ask for help from, and provide help to, others. **You may not, of course, blindly copy solutions from one another (or from anywhere else) or simply write code for someone else,** but you can certainly help each other debug, give plenty of suggestions and hints, *explain* why things work or don't work, etc.  
Read the full policy at: [Full Collaboration Policy](https://docs.google.com/document/d/1WyzL3qvKLrC1UCRf178b_wYWQmEZlhDObFNFb79U63I/edit?usp=sharing)

**Lab Objectives**

To gain experience with:

|  |  |  |  |
| --- | --- | --- | --- |
| * input | * output | * string variables | * using a terminal |

**C9 and Terminal Refresher**

1. Recall, cd means change directory. The directory you wish to change into must be specified with the command and the specified directory must be within your current directory. For example, if cs010\_practice existed immediately below our current directory we would say:  
   cd cs010\_practice
2. The home command will take you to the top level of your workspace tree, from anywhere in your workspace.
3. To compile a program we use g++, when successful g++ creates an executable file named a.out. We must be in the same directory as the file we wish to compile. So first change the directory all the way to where you have your C++ source code file. Then we use g++:  
   g++ lab2.cpp
4. To execute a program we use the run command. The run command can be called in several ways, and we can use run --help for help identifying how to call the run command. Again, like g++, we must be in the same directory as our executable file. Our most basic usage simply executes our executable file:  
   run a.out
5. To create a file, we can use File->New File, but a more simplistic approach is to right click on the folder you wish to create a new file in and select new file.
6. To save, use ctrl+s or File->Save. Always save before compiling or downloading your file.

**Exercise 1: What is Wrong with My Code**

For your first exercise you will need your survey ID. First find your survey ID on CoLe or iLearn under your grades. Once you have your 4 digit survey ID, the study is at: <http://goo.gl/KmRaFp> (Study will go live Sun after 9pm.)

**Exercise 2: Hello, You!**

You will be writing a simple program involving both input and output. The program initially prompts the user for his or her name. After the prompt the program should acquire a name through input. Once the name is acquired, the program asks the user a question about nice days. After which, the user will enter how many nice days he or she wants. Lastly, the program outputs a response back to the user; the response will use all the provided inputs, and double the number of nice days!

**Setup**

1. Create a file named lab2.cpp within the proper directory for Lab 2.
2. Fill the file with the contents of your lab1.cpp file.
3. Delete everything between main's left and right curly braces, { and }, except the return 0;
4. Make sure the assessment header in your new file is properly filled in.

**Think About the Problem**

1. With a program we often design an algorithm to solve our problem. However, with shorter programs we can often bypass a general algorithm, and simply use comments within our source code to lay out our problem before we program.
2. Based on the description of the problem, what are the tasks the program must do?
   1. each task should be a comment
   2. recall comments start with //
   3. each comment should be an English description of the task, not C++ code
3. Certain tasks may require storage of values, we store values in variables.
   1. Take a moment to identify how many items you wish to store, based on your tasks.
   2. Create a comment with an English description for each item you wish to store.
   3. Now, think about the type of variable which is required to store each of the identified items.
   4. Depending on the program, you may use different datatypes for different variables.
4. Now, within your comments is a clearly outlined program. No C++ code should exist within your source file; except the basic starting structure of the main function, think of it as a C++ code template.

**Program Your Solution**

1. Now, on the line following each of your comments, write the C++ code that performs that task: this allows a reader to read and think about the comment, and then check (or skip) the actual implementation.
2. You will often need to to check an example program run to get your output to be as expected. We will generally provide some example runs in the specification (e.g. below), but we also provide executables for some assignments in the cs010\_programs area.

**Example Execution** (User input has been **bolded and underlined** to help differentiate typed input from program output.)

user@cs:~/.../lab02 $ g++ lab2.cpp

user@cs:~/.../lab02 $ run a.out

Good day, what is your name? **Tom**

How many nice days would you like, Tom? **4**

Hi, Tom, have 8 wonderful days!

**Submission Instructions**

Submit your completed .cpp file to [R’Sub](https://galah.cs.ucr.edu) for testing, feedback and grading - see [here](https://docs.google.com/document/d/17NUZJlKGcO7r8Q8mzfGjN1EhAFlMiqQU4erD9weoRUw/edit#bookmark=id.5hzrzg5c9tun) for instructions.