CSE 100, CIDSE, Fall B 2013
Lab #1, Arizona State University
Introduction to variables, input and output statements

Name (Last, First): _____

Pre-Lab - Please complete this section before your lab session and submit.

Purpose: This pre-lab will introduce you to variables and assignment statements. You will need to be familiar with these three topics in order to do the lab.

Variables:

Note: If you already know what variables are and how to declare and initialize them, then just look at the bold words and skip the rest of this section.

Variables are place holders for data used in programs. Variables have two parts: a ***data type*** and a ***variable name***. The data type tells the computer what kind of data should go into the variable (and what size the data is). Common data types are: ***bool*** (stores "true" or "false"), ***int*** (stores integers), ***double*** (stores numbers with decimal parts), ***char*** (stores a single character, like 'Q'), and ***string*** (stores many characters, like "She sells seashells by the seashore.").

The second part is the variable name. A variable's name is what we use to access the data stored inside. All variables must be unique in C++. Variable names can be made up from letters (upper- and lower-case), numbers, and the underscore symbol ("_"). However, variables can't begin with a number. No other characters are allowed as part of a variable name, *including spaces*. C++ is case-sensitive, which means that gasMileage, gasmileage, and GasMileage all count as different variable names. Try to avoid variables with similar names, though, as it's easy to get them confused. Here are some examples of declaring variables:

```
char letterOfTheAlphabet;
bool ToBeOrNot2Be;
double cost_of_item, tax_on_item;
```

In addition to ***declaring*** variables (telling C++ their names and what data they'll hold), you have to ***initialize***, or ***define***, them. Initializing a variable is giving it an initial value. It's good habit to give variables an initial value right after you declare them, otherwise it opens the door to all kinds of cryptic problems. Here are some examples of initializing variables (modifying the example above):

```
char letterOfTheAlphabet = 'Q';
bool ToBeOrNot2Be = true;
double cost_of_item, tax_on_item;
cost_of_item = 9.95;
tax_on_item = 0.87;
```

Note that you can give a variable an initial value after declaring it (as with *cost_of_item*), or as you declare it (like *letterOfTheAlphabet*). Since you can only declare a variable once, the second kind of initialization only works once. If you want to give a variable another value, you'll have to use a normal statement.

Statements:

Note: If you are already familiar with C++ statements and you know how cin and cout work, feel free to skip this section.

A ***statement*** is a complete bit of code that the computer executes. At their most basic level, statements are a line of code that ends with a semicolon (";"). Statements are kind of like sentences. There are a bunch of different kinds of statements, depending upon what the statement accomplishes. The simplest kind of statement is an ***assignment*** statement. This comes in the form "*variable = something;*", where *variable* is a valid variable name, and *something* is a value (or something yielding a value) that we want to put in the variable. Things like this are valid assignment statements:

```
cost_of_item = 9.95;
tax_on_item = cost_of_item * taxRate;
ToBeOrNot2Be = true;
```

Another two kinds of statements are ***input statements*** and ***output statements***. C++ uses a special way of getting input from and sending output to the user. Don't worry, you just have to learn how to use it, not how (or why) it works. For now, just pretend that gerbils make it all work behind the scenes.

You will use ***cout*** to give data to the user and ***cin*** to get data from the user. An example of using cout would be:

```
cout << "Good morning, user #" << user_number << endl;
```

If *user_number* had the value 54321, the user would see:

```
Good morning, user #54321
```

As you can see, anything inside double quotes is simply output word-for-word. Variables have their values printed out, and "endl" is like hitting Enter (the cursor will go down to the next line). You use "<<" to tell *cout* about each thing it should print for the user. From the example above, you can see that you can chain these together to have *cout* print multiple things at once.

Input works similarly to output. Using *cin*, you can get information from a user, like this:

```
cin >> userAge;
```

Upon executing this line, the program would pause, the user would see a blinking cursor on the console, and they could type something in. When they hit Enter, whatever they typed in would get placed into *userAge*, and the program would continue running. Note that with *cin*, the brackets point toward the variable (there's a hint for which way they should go!). Yes, you can also chain inputs with *cin*, like this:

```
cin >> userAge >> userHeight >> userWeight;
```

but be careful, because this can lead to strange behavior, both from the user (who might get confused) and from the program (which might end up with unexpected input).

Warning: You can't mix output and input in the same statement!
This: `cout << "Please enter your age:" >> userAge;`
This: `cin << "Please enter your age:" >> userAge;`
And this: `cin >> userAge << endl;`
Are all bad! They will all cause C++ to have a fit, and will make your TA bang their head on the table. The right way to get the user's age from a prompt would look like this:

```
cout << "Please enter your age:";
cin >> userAge;
```

Part I- Pre-lab Questions – Submission Required and this will be graded
Include your solutions as comment lines in the Lab1.cpp program that you will develop in part II

Directions: Answer these questions to achieve enlightenment (and to get full credit for your lab !).

Lab Letter: ____ Name (Last, First): _____

1. Name five data types and what kind of data are they used for.

2. Write the declaration statement for a variable of data type *char* with the variable name *gender* and the initial value 'M'.

3. Write next to each statement whether or not it is correct. If it's incorrect, write a correct version of the statement.

char key = c;

*rate * hours = pay;*

record = "Dental"

*uselessNumber = 3.2 * 6;*

cout >> the number of squares is >> num_of_squares;

4. Arithmetic Operators and Operator precedence rules

Expected Output Observed Output

```
#include <iostream>
#include<cmath>
using namespace std;
1 int main()
```

```
2 {  int someInt,                                          _____________   _____________
3        w = 5, x = 9, y = 2, z = 7;
4    char someChar = 'A';                                  _____________   _____________
5    cout << "operator precedence \n";
6    cout << (x-1) / (x-w) * y << endl;                    _____________   _____________
7    cout << (x-1) / ((x-w) * y) << endl;
8    cout << static_cast<double>(x) / y << endl;           _____________   _____________
9    cout << x / y << endl;                                _____________   _____________
10    cout << (w + x % 7 / y) << endl;                     _____________   _____________
11    cout << (abs(y - w) + sqrt(x)) << endl;              _____________   _____________
12    someInt = someChar;                                  _____________   _____________
13    cout << someChar << "  "                             _____________   _____________
14         << someInt << endl;                             _____________   _____________
15
16    return 0;
17 }                                                       _____________   _____________
```

Lab 1: Part II- Submission Required

Description: Housing expenditure calculation program

Write a C++ program that asks the user to enter his/her monthly costs for each of the following expenses

- Rent or mortgage
- Phones
- Utilities
- Cable

The program should then display the total monthly expense. Make sure to display the total cost using two decimal points **(see program, 3-16 in page 116 for a number formatting example)**

Name your program as Lab2.cpp and upload your program into the dropbox lab1 basket.