CS121 SI – Week 1 – Pt. II Solutions

Concept & Syntax Questions:

1. Give an example of declaring a variable and another of initializing a variable. Is it okay to use a variable that's initialized but not declared? Declared but not initialized?

   ```
    long my_var;  // Declaration
   my_var = 6;   // Initialization
   ```

   Initialized but not declared: No. The compiler will say the variable does not exist
   Declared but not initialized: No. The compiler usually assigns a junk value to variables on default. Lesson to learn here is to always initialize your variables once you declare them.

2. What is the output of the following code and why? Assume that there is an integer called *a* that is equal to *4*.

   ```
   if(3 == a)
      if(4 >= a)
      {
          cout << "Your number is either 3 or 4";
      }
   else
   {
       cout << "Your number is not 3";
   }
   ```

   There is no output. The reason for this is because, without specifying the start/end (with curly braces) of the first if-statement, the else statement attaches itself to the closest if statement. This is famously known as the dangling else problem.

3. What is an escape sequence? Give three examples of them and what each of your examples do.

   An escape sequence is a special series of characters that perform a special task.

   Examples: '\n' (new line), '\t' (tab), '\a' (alarm/bell to make a beep noise)

4. What is the output of the following code (in Visual Studio) and why is that?

   ```
   cout << sizeof(float) << endl;          4

   cout << sizeof(short) << endl;          2

   cout << sizeof(int) << endl;            4

   cout << sizeof(double) << endl;         8
   ```

   Each number is the byte-size per datatype inside of *sizeof*. That's what sizeof does.
   Note that each size can vary, depending on the compiler and OS bit size (32 or 64 bit).

5. Give an example of using camel case and using underscores for variable identifiers. Is one better than the other? Explain.

   double thisIsCamelCase;

   std::string this_is_underscores;

   Without bias, neither is better as both are fairly readable. What matters most is consistency. If you choose one for doing one thing (e.g. labelling function names with camel case and variables with underscores), use it for that specifically.

6. If we assign an integer *num* to *4.85*, what is the value of *num?* Explain. Also, if we wanted to print the (true) result of *3 divided by 4* with *cout*, how could we do so?

   int num = 4.85;

   cout << "This is my number: " << num << endl; // this prints out This is my number: 4

   cout << "3 / 4: " << (3.0/4) << endl;   // or 3/4.0

   For both examples, integers are not designed to hold decimal places or fractions. C++ truncates (cuts off) the decimal place automatically.

7. What is the modulus operator, and in what are two situations where using the modulus operator is considerably useful?

   The modulus operator returns the remainder of integer division. An example is *3 % 4*, which returns *3* as *3* divided by *4* is *0* remainder *3* (think back to third grade).

   The modulus operator is considerably useful when finding if a number is even or odd (use *num % 2* as all evens are divisible by *2*), and when generating random integers (range-based calculation).