Name: _____ **Section:** _____ **Grade:** _____

Answer these questions while viewing the assigned videos. ***Not sure of an answer?*** Ask your instructor to explain ***at the beginning of the next class session***. You can then fill in your answer, still for full credit. (But no fair doing that unless you attempted the question first.)

Videos for this quiz: www.rose-hulman.edu/class/csse/binaries/C-Videos/session1/

Video 1: Why C? (3 minutes, 39 seconds)

1. The programming language C is more than 30 years old. ***True*** or ***False***? (Circle your choice.)

2. All the following departments use the programming language C in some of their courses: ME, ECE, Math and CSSE . ***True*** or ***False***? (Circle your choice.)

3. True or False: Many programming languages, e.g. Java, have a syntax that is, at least in part, similar to that of C. ***True*** or ***False***? (Circle your choice.)

4. Which provides closer access to the hardware? **C** or ***Python***? (Circle your choice.)

5. Which has more built-in higher-level language features? **C** or ***Python***? (Circle your choice.)

6. Which language's code generally runs faster? **C** or ***Python***? (Circle your choice.)

7. Which language provides code that is more *portable*, that is, can be more easily run on various kinds of computers? **C** or ***Python***? (Circle your choice.)

8. What is the difference between an ***interpreter*** and a ***compiler***?

   - Interpreter:

   - Compiler:

9. Python code is ***interpreted*** or ***compiled***? (Circle your choice.)

10. C code is ***interpreted*** or ***compiled***? (Circle your choice.)

11. What does ***static typing*** mean?

12. There are two primary advantages of a language having *static typing* (as in C).
    - One is that compilers for statically typed languages (like C) can, in general, produce smaller code that runs faster than the code produced for dynamically typed languages. That's because static typing gives the compiler more information than dynamic typing does – in static typing the compiler knows the types of the variables and can make good use of that information.
    - What is the ***other*** primary advantage of a language having ***static typing***?

13. There are two primary advantages of a language having *dynamic typing* (as in Python).
    - One is that dynamic typing allows for more flexible code – for example, the same code might be able to handle integers, floats and perhaps even as-yet-unimagined numeric types, with the details of what the code does being determined at run-time by the actual type of the variables encountered.
    - What is the ***other*** primary advantage of a language having ***dynamic typing***?

Video 2: Strings and Printf (3 minutes, 57 seconds)

14. **True** or **False** (circle one): The literals **"C"** and **'C'** mean the same thing in C.

15. For this problem, the following elaboration from the video may be helpful:

    - In Python, a variable can refer to an object of one type at one point and an object of another type later in the run. In C, each variable must be associated with a type when the variable is first mentioned. We call that ***declaring*** the variable, as required by ***static typing***.
    - The notation for declaring a variable is the *type/name* pattern, as in these examples:

      ```
      int count;
      float temperature;
      double gravitational_constant;
      char middle_initial;
      char* my_name;
      ```

      The above examples illustrate the five primary types that we will use in C: integer, floating point numbers (*float* for single precision, *double* for double precision), character and string (note the asterisk in the declaration).
    - Each statement (which is NOT the same as "each line") in C concludes with a semi-colon.
    - You ***assign*** a variable a value in C just like you do in Python (except strings are different, as we'll see later):

      ```
      count = 0;
      temperature = 98.6;
      middle_initial = 'A';
      ```

    - In the code you write for me, separate the variable's ***declaration*** from the variable's ***assignment*** (the video combines the two, which is fine for experienced C programmers but not for you).

[Note: This and all code hereafter is to be written in C unless directed otherwise!] Now:

a. Write statements that ***declare*** an integer variable **k** and a floating point variable **gpa**.

b. Alena has a *perfect* grade point average. Write a statement that sets the variable **gpa** that you declared above to her (perfect) grade point average.

c. Suppose that variables **`price`** and **`year`** have been declared. Suppose further that these variables have been and assigned values that are the price and year, respectively, of a particular car (for example, a 2011 Honda Civic). Write a statement that prints the values of those two variables. Make any reasonable assumption about how many digits and (for the price) what precision is appropriate.

d. Write *two separate* statements (unlike the single, combined statement shown in the video) that:

- declare a variable **`fahr`** and
- assign it the value returned from calling the function **`convertCtoF`** in the video, where you send that function the value **`20.0`**.

Video 3: If and While (7 minutes, 12 seconds)

16. In ***Python***, what notation do we use to ***group*** a collection of statements?

17. In ***C***, what notation do we use to ***group*** a collection of statements?

18. Write the following Python statement in C. (Use the space to the right.)

```
if n > 0:
    print "Pos"
elif n < 0:
    print "Neg"
else:
    print "Zero"
```

19. **True** or **False** (circle one): Microsoft and many other companies require, as part of their coding standards, that the programmer uses curly-braces for the bodies of ***`if`*** statements, even when the body is a single statement. (Hint: The answer is *True*, and our coding standard also requires the curly-braces for ***`if`***, ***`while`*** and ***`for`*** statements.)

20. In C, _____ is like **False** in Python and _____ is like **True** in Python.

21. Write the **C** operators for:

and: _____ ***or***: _____ ***not***: _____

22. Suppose that **grade** is a variable that stands for a grade on an assignment, and suppose that the professor of the course uses the standard 10-point grading scale (90 and up is an A, etc). Fill in the blanks appropriately (don't forget the various punctuation):

```
if __   ____________________________________________________   __   __
    printf("The grade is a B.\n");
_____
```

Video 4: Scanf (2 minutes, 49 seconds)

23. For *scanf*, the following elaboration from the video may be helpful:

- The ***scanf*** function in C gets input from the console, just like ***input*** in Python.
- The ***scanf*** function does NOT include the prompt. You have to do a ***printf*** of your own to print the prompt. In our environment, ***printf***'s that appear just before a ***scanf*** need to have the "buffer flushed". This is what the ***fflush*** accomplishes.
- The ***scanf*** function requires that you say the type of the input (and doing it wrong creates a very-hard-to-find error).
- Both ***printf*** and ***scanf*** allow **%i** as an almost-identical synonym for **%d**. I'll use **%i** instead of **%d** in all my examples, since ***i*** for ***integer*** is easier to remember.

Here is an example, in Python (on the left)| and C (on the right):

```
n = int(input('Enter an integer: '))
```

```
int n;
printf("Enter an integer: ");
fflush(stdout);
scanf("%i", &n);
```

Now, write the statements that would declare a variable for, prompt for, and input from the user a floating-point number (say, the weight of a certain book). Reminder: **%i** for integer, **%f** for floating-point.

Video 5: Menu (9 minutes, 54 seconds)

There are no quiz questions from this video, but you will implement a similar function in your first C assignment in Eclipse.

If you have other questions about these videos, ask them at your next class session!