CSI 201
console, types, variables and arithmetic

1. To output something to the console we use `cout`. We attach new things to `cout` with `<<`. `endl` stands for end line and it is something we can attach to cout to work as a "enter" key when typing into the console. So let's look at our first line of code!

   ```
   cout << "Hello World" << endl;
   ```

2. Try to output something different to the console:

3. What is the console output of the following code:
   ```
   cout << "hi";
   cout << "everyone";
   ```

4. Anything inside " " is called a string literal. We will use these a lot to communicate with the user.

5. Can you find some errors in the following code:
   ```
   cout << please" << endl;
   cout << "help me fix this"
   ```

6. Okay, so let's talk about how to get stuff from keyboard input. We use cin with variables. An example might be:
   ```
   string your_name;
   cin >> your_name;
   ```

7. We can output those variables just like our string literals. Remember, how we attach things to `cout`?
   ```
   cout << "I see your name is " << your_name << "." << endl;
   ```

8. The line above has 3 things attached to the console output statement. We use a string literal, a variable and then our special `endl`. Try making your own variable and constructing an output statement like the one shown above.

9. It turns out there are other types of variables as well. `string` is called a variable type. Another common type is an `int` which stands for integer. These are numbers from -2,147,483,648 to 2,147,483,647. This includes all the numbers in between (but does not include numbers with decimals - we will use another type called `double` for that). It is important to understand that `int` in C++ has limits. Remembering roughly 2.1 billion is useful. We can type those integers into our code literally (integer literals) by using code like the following:
   ```
   cout << 555 << endl;
   cout << "3 / 2 is " << 3 / 2 << endl;
   ```
10. Can you identify the different types of literals in the statement above? There should be a string literal and more than one integer literal.
11. Also, if we were to actually use the code above, what do we expect as output? The answer might surprise you. Let's discuss and then experiment.
12. Variables start with letters or underscores (_) and continue with letters, numbers and (_) but they must avoid C++ keywords (like int, for, etc). We will learn more of these keywords throughout the semester.
13. As part of today's discussion we also explored the zybooks environment and how programming in C++ works. Specifically you need to know the following:
    (a) We write source code in a high level programming language called C++.
    (b) We use a compiler to translate this source code into a machine readable executable. The compiler translates what we write in C++ into an actual program that a computer can run.
    (c) The compiler also operates as a kind of oracle. It will find mistakes in your sentences and try to tell you where you went wrong. Sometimes the output is somewhat cryptic. The compiler tells us the line where a mistake may have occurred but the results must still be interpreted and corrected by us. And the more mistakes we make, the more you will see the compiler errors and the easier it is to understand what the compiler is trying to tell us. The compiler will not find errors that will occur because of failed logic or things that can break when you run the program. The compiler finds errors in the process of its translation. Its job isn't to find your errors, but rather, it does this in the process of its job - which is translating the source code into the executable that the machine can read.
    (d) If your code compiled correctly into an executable, we can then run the program. It runs from the executable, not your C++ source code. So, changing the source, requires a new compile. Running the program is typically what happens when you double click an icon on your machine. In our zybooks, the compile and run often happens in one step. In most C++ interactive development environments (IDEs) this is also the case. However, it is important to understand that compiling the code and running the code are two distinct steps that require two different types of resources. One requires source code. The other requires machine code.

(e) In zybooks, input from the console is not interactive, but in a lot of other environments it might be. This changes the way we think about input and output slightly. We will highlight this change in other places throughout this course.