CSI 201
for loops!

1. Goals for today: for loops, strings, getline!!
2. Example #1: Repeating a task many times. Output a user input number of stars!

```
for(int i = 0; i < 10; i++) {
    //this task happens 10 times!
}
```

3. Can you output 100 stars using a for loop?

4. The initial number in a sequence is 3. Each following number is 2 higher. Start with a variable a with a value of 3. Write a for loop that increases the value of a by 2. Let this loop run 100 times. After the for loop, output this number.

5. What value should we get?
6. Can we do this for other types of sequences?
7. `for` loops are great for repeating tasks certain numbers of times. They're also great for walking through lists of things. We'll look at that later. For now, we'll simply use them to repeat a task. You can use them to build ammortization tables, simulate physics, run loops in games and crunch numbers on any variety of tasks (like digits of pi from the last lab!).

8. Strings have the ability to give us their length. Given a string s, we can find the length with:

```
string s = "Ramsey";
cout << s.length() << endl;
```

9. We might use this to output stars in place of a password. If the user's password is 10 characters long, we can output 10 characters. Imagine you have a string variable called password. We might write the code:

```
for(unsigned i = 0; i < password.length(); i++) {
   cout << "*";
}
cout << endl;
```

10. Here you'll notice a new type `unsigned`. This type is used for counting and is often the return value of a function which cannot be negative (like the length of a string).

11. If we use an int in this loop for i, then we will sometiems get a compiler warning/error. So, in zybooks, we'll use unsigned in for loops and conditions that must interact with an unsigned function or variable.

12. What does the trailing `cout << endl;` do? Is it required?

13. We could do that in place of `cout << password;` to obscure a password output.

14. Lastly, there are other ways to get input from the user. So far we have been using `cin` with the `>>` operator. However, we can also use a special piece of code called a `getline`. This will read in an entire line. Unfortunately, `getline` and `>>` do not work together as gracefully as desired and extra care will be required in those cases. For now, let's focus on getline by itself. Let's say we want the user to enter their full name. `getline` will be capable of reading that entire line of text into a variable for us.

15. Sample code:

```
string fullname;
getline(cin, fullname);
cout << fullname;
```

16. Let's do some demo's and show the difference between: "Doc Ramsey" entered on one line and then on two lines.

17. You can now imagine asking for a username and password. We can use getline to read the username from one line and the password from a second line. Then we can report the username back to the user and using the for loops from above, we can report a number of asterisks based on the length of the string given to us.

18. For practice: Write a for loop that outputs a box of stars based on the user's input. For example, if the user enters 2, then you print 2 stars on the first line and 2 stars on the second line. If the user enters 3, 9 stars total should be printed (3 on line 1, 3 on line 2 and 3 on line 3). See if you can make this happen. Hint: you might want to use a for loop inside another for loop!