CSI 201
practice, cerr, if-else

1. Goals for today: Practice, if statements, else statements!

2. I wanted to make sure you saw the framework that our programs are going into again:

```
#include<iostream>
using namespace std;
int main() {
    //your code goes here
}
```

3. With that out of the way, let's get to our first practice of the day. I'll provide some sample code as well, so that you have some things to look at. Sample code from the end of last class, first:

```
double a,b;
a = 0;          //these are new, these are initializations
b = 0;
cin >> a >> b;
cout << a * b << endl;
```

4. Create a program that reads in the value of a dinner bill. Then, the program will read in a percentage for the tip. Something like 10 for 10% or 10.5 for 10.5%. Lastly, compute the final price of the dinner bill after giving a tip of this size and paying the bill. Sample input: `50 10` Sample output: `You should pay $55`
Recall this 55 comes from 10% of 50 plus 50 itself (although there might be other ways to compute this value).

5. Great, let's give this a run in zybooks!

6. Is there anything we can do to the user input to make things break? At your tables, everyone take a minute or two to come up with a couple broken examples.

7. Okay, so what we need to be able to do is check the values of the user input to make sure they are not broken. These happens through the use of something called an `if` statement. Example code snippet:

```
double velocity;
cin >> velocity;
if(velocity < 0) {  //CHECK FOR BADNESS!
   cerr << "ERROR velocity=" << velocity << endl;
}
cout << "Velocity: " << velocity << endl;
```

8. Some new things show up here! Let's first notice that there is a cerr statement. `cerr` works just like `cout` except that it operates in a different way with the console. For now, it is important to know that we use it for errors rather than regular output, but that otherwise we can think of it as exactly like cout.

9. The biggest new piece is the if statement! if statements have a few important parts. First, the actual word `if` itself is a keyword in C++ and it is going to signify that we want to test something. The thing we want to test appears in parens () just after the word if. We can do all sorts of tests here, but most often we will use these tests in conjunction with `<,>,<=,>=,==, and !=`. Make sure you understand what is different about each of those and what each means. These are called relational operators and they are a good way to test two values with one another. The last part of the if statement appears between curly brackets {}. The code inside these curly brackets is run if and only if the condition tested is true. So in our code example, if the velocity that the user entered is actually less than 0, then the cerr statement will run. Otherwise, that part of the code is skipped over!

10. Write an if statement for your tip code to see if you can improve it.

11. Sometimes, you want to do different things if the result is true. For example, in the code given above, it would be nice if the second cout doesn't run if the error statement runs. In this case we really want a branch in the code rather than to just skip over a line of code. In English we might say something like: if this is true, then do this, otherwise, do something else. This is how if statements get their name. It is also how the `else` part of an if statement gets its name. Together, we can write `if-else` statements for our velocity situation. Sample code might be similar to that written above:

```
double velocity;
cin >> velocity;
if(velocity < 0) {  //CHECK FOR BADNESS!
   cerr << "ERROR velocity=" << velocity << endl;
}
else {  //if all is good then let's do this
   cout << "Velocity: " << velocity << endl;
}
```

12. Okay, time to practice on your tip program. There were some bad user inputs right? Well, let's write some code to make this better! Get to work.

13. If you have extra time, let's revisit some of our polynomial math! Imagine you had the user give you three inputs: a,b, and c and you treated it as: $a * x^2 + b * x + c$ In some situations, this thing will not factor with real roots. It only factors with real roots if $b^2 - 4ac$ is bigger than or equal to 0. So write some code that describes if a given set of inputs, a,b, c will have real roots or not.