Code Layout

The goal of code layout/formatting is to show logical structure

Good layout is shows intention, is consistent, improves readability, and withstands modification.
Which is best?

A) \( \text{for}(\text{int } i=0; i<\text{str.length}(); i++)\) {
B) \( \text{for} (\text{int } i=0; i<\text{str.length}(); i++) \) {
C) \( \text{for} (\text{int } i = 0; i < \text{str.length}(); i++) \) {
D) \( \text{for} (\text{int } i = 0 ; i < \text{str.length}(); i ++) \) {
E) \( \text{for}(\text{int } i = 0 ; i < \text{str.length}(); i ++) \){
Which is better?

A) for (int i = 0; i < args.length; i++)
B) for (int i = 0;
        i < args.length;
        i++)
C) Both are fine
D) Both are lacking
Which is better?

A) if (game[i][index] != c)

B) if ( game[ i ][ index ] != c )

C) Both are fine

D) Both are lacking
Which is better?

A) \( \text{char}[][][] \text{ game} = \text{new} \text{ char}[3][3]; \)
B) \( \text{char}[][][] \text{ game} = \text{new} \text{ char}[3][3]; \)
C) Both are fine
D) Both are lacking
Hmmm...

- I like spacing operands like the following:
  \[ \text{int } x = a + b + c + d + 17; \]

- But in the below, I personally prefer the second option:
  \[ \text{data}[i][i] = \text{data}[i-1][i-1]; \]
  \[ \text{data}[i][i] = \text{data}[i-1][i-1]; \leftarrow \]
Are you familiar with the ternary operator?

```java
if (a) {
    x = b;
} else {
    x = c;
}
```

```java
x = a ? b : c
```

`boolean_expression ? option_true : option_false`
What is wrong with this?

```c
int x = b > 10 ? c : d;

int foo = a + (b > 10 ? c : d) + e;

= a + x + e
```
Which is better?

A) `int parenthesis;
   parenthesis = 0;`
B) `int parenthesis = 0;`

C) Both are fine
D) Both are lacking
Which is better?

A) int paren = 0, eqnLength = eqn.length();
B) int paren = 0;
int eqnLength = eqn.length();

C) Both are fine
D) Both are lacking
What is wrong with this?

\textbf{aFunction}(j, k); j++; k++;
Which is better?

A) if (three) {
    System.out.println("Valid");
} else {
    System.out.println("Invalid");
}

B) if (three) {
    System.out.println("Valid");
} else {
    System.out.println("Invalid");
}

C) Both are fine  D) Both are lacking
if (three) {
    System.out.println("Valid");
}
else {
    System.out.println("Invalid");
}

System.out.println(three ? "Valid" : "Invalid");
Which is better?

A)
```java
if (three) {
    System.out.println("Valid");
} else {
    System.out.println("Invalid");
}
```

B)
```java
if (three)
    System.out.println("Valid");
else
    System.out.println("Invalid");
```

C) Both are fine
D) Both are lacking
Which is best?

A) if (prev_type==type&&type!=1&&type!=2) {
B) if (prev_type == type && type != 1 && type != 2) {
C) if ((prev_type == type) && (type != 1) && (type != 2)) {
D) All are fine
E) All are lacking
What is wrong with this?

```java
if (c.equals("*") ||
    c.equals("/") ||
    c.equals("+") ||
    c.equals("-")) {
```
If statements

What is wrong with:

```java
if (winner == true) {
    aFunctionCall(arg1, arg2);
}
```
Which is better?

A
if (count != 3) {
    return false;
}
return true;

B
if (count == 3) {
    return true;
}
return false;

C) Both are fine

D) Both are lacking

return (count == 3);
To do for Tuesday

- Read Ch. 22 of Code Complete (Testing).
  - Reading questions will be up for both Ch. 31 and Ch.22
- Learn JUnit 4
  - See: https://youtu.be/b5W8bR4y-g4
- Write tests for the class that you wrote.
  - See how exhaustive a test suite you can generate
  - Check them into your project in SVN.
  - Feel free to fix bugs in your code, but I’m more concerned with writing a solid test suite.