Pop Quiz

Define a function PrintThrice(s) that has a string argument, and prints its argument three times on one line, and then goes to a new line. For example,

```
PrintThrice ("Ho! ");
```

prints:

Ho! Ho! Ho!

```
void PrintThrice (string s) {
    cout << s << s << s << endl;
}
```

```
    cout << s;
    cout << s;
    cout << s << endl;

    for (int counter = 1; counter <= 3; counter++ ) {
        cout << s;
    }
    cout << endl;
```

Exam I

Coming in one week (2/17).

Roughly 15 questions.

1/3 short answer, 1/3 code reading, 1/3 code writing.

LCR 1-4, TCS 1-4.

I've posted on the schedule page a set of Review Questions and an Answer Key to them. The idea is: after you have studied, try to answer the Review Questions as though you were taking the exam. Only after you have done this, check the Answer Key. If you have any questions about the answers to the Review Questions, or any of the other material, ask in the review sessions in you lab.
You can have a 1-page single-sided "cheat sheet," either hand-written or printed with at least 11pt font.

Conditional Execution:

A major process in almost any non-trivial program is making decisions. We do something only if some condition is true. Or we do one thing or another depending on whether some condition is true or false.

In the simplest case, conditions are things like relations between values (==, !=, <, >, <=, >=). We can compare two values of the same type this way.

Conditional Statements:

One-branch conditional:

```
if ( <condition> ) {
      <statements>
}
```

Two-branch conditional:

```
if ( <condition ) {
      <true-branch statements>
```

```
} else {
    <false-branch statements>
}
```

Arrange your conditions so that the logical structure of your program is apparent to the eye.

Chained-conditional:

```
if ( <cond 1> ) {
    <true branch 1>
} else if ( <cond 2> ) {
    <true branch 2>
} else if ( <cond 3> ) {
    <true branch 3>
} else {
    <all false branch>
}
```

Nested conditional:

In a conditional such as
if (<condition>) { <true branch> } else { <false branch> }

the statements in the <true branch> and <false branch> may include other conditional statements.

Boolean values:
C++ has a type called bool, which has two values, true and false. These are

effectively equivalent to 1 and 0, respectively. They are truth values.

bool is a "first-class citizen" in C++. Just like any other type in C++ (int, double, char, string,…), you can have bool variables, bool arguments, bool-valued functions, and bool operators.

A relation like "N < 0" compares two numbers and returns a bool value (true or false). An if-statement, while-loop, or a for-loop (in the <continuation> part) expects a boolean value.

C++ provides a number of logical operators:

! X == the negation of X
X && Y == true if and only if both X and Y are true
X || Y == true if and only if either X or Y or both are true (inclusive OR)

You have to get used to working with these boolean operators.

(! (N < 0)) == (N >= 0)
(! (X && Y)) == ((!X) || (!Y)) // DeMorgan's Law
(! (X || Y)) == ((!X) && (!Y)) // DeMorgan's Law

Warning: Do not confuse = (assignment operator) and == (equality relation or condition).

X = 1; // puts 1 in X and also returns it.
X == 1 /* compares the value of X and 1, and returns true (1) if they're the same and false (0) if they're not */

For various (not very good) design reasons in C++, the C++ compiler cannot tell when you use one of these but meant the other.

```
int X = 0;
if (X = 0) { cout << "X is zero\n"; }
else { cout << "X is not zero\n"; }
```

The above prints "X is not zero."

```
int X = 0;
if (X = 1) { cout << "X is one\n";
} else { cout << "X is not one\n";
}
```

The above prints "X is one."

Desk-check your programs!

Recursive Functions:
Recursive functions in effect, put input into their own In Box.
But it's a little more complicated than that.

Analogy: suppose we have an office that looks stuff up (looks up a word in dictionary or wikipedia), and summarizes it. This office might need to make a request of another that does the same function (but with different workspace).