CS211 Review: Topics to Know

This packet lists some of the topics you should know prior from entering this course. It does not cover ALL the materials that have been covered in the previous semester.

Ask plenty of questions during lab for the topics you do not understand.

Ch 1: C++ basics: variables, expressions, = statements, console input/output, program style, libraries (i.e. cmath) and namespaces

Variable Names (Identifiers)

- Must start with either a letter or an underscore (_). The rest of the characters may comprise of letters, digits, or underscores.
- Are case-sensitive. Meaning, although two variables may consist of the same letters, they are distinct from each other if their upper/lower casing is different.
  For example, *rate*, *RATE*, and *Rate* are considered as three different variables.
- You are not allowed to use keywords/reversed words to name your variables. For example, **int** is a reserved keyword, thus you are unable to do this:

  Example) string int; // not a valid variable name

The entire list of C++ keywords are available in Appendix 1 towards the back of the book.

Variables

- A variable must be **declared** (given a type) before it is used. "declaration"

  Example:
  ```
  int n;
  string word;
  double fahrenheit, celsius;
  bool isPrime;
  ```

Assignment Statements (=)

- Contains a variable on the left-hand side (**LVALUE**), an expression on the right-hand side (**RVALUE**). The value of the right-hand side is **assigned** to the variable on the left-hand side.

  Example: `n = 10;`
- An expression may be a variable, number, Boolean, etc… Expressions are anything that evaluates to a value.
- Assigning a value to a variable for the first time is called **initialization**.

  Example:
  ```
  int n;
  n = 10;
  ```
  Example: `int x = 5;`
- As discussed in lab, where variables m and n are declared as integers, **m = n = 2;** is a valid statement.
- Extra Credit: Is **(m = n) = 2;** a valid statement? Why/why not?

Question: Why is **cout << (m = 2);** a valid statement, but **cout << m = 2**; causes an error?

You Should Know:

- Escape Sequences ("\n")
- Arithmetic Operators and Expressions (Addition, Subtraction, Multiplication, Division) and Integer & Floating Point Division
- Casting (double), or double(___)

- cin/cout
- Libraries and Namespace
- Pre-increment/decrement operators and post-increment/decrement operators

Question: Write a complete program that asks the User for a Celsius value. Convert the User's input into Fahrenheit and output it to the console. The conversion formula from Celsius to Fahrenheit is: $F = \frac{9}{5}C + 32$

Ch 2: flow of control: boolean expressions, if, if/else, loops

Boolean Expressions

- Expression that evaluates to TRUE or FALSE.
- You should master: ==, !=, !, <, <=, >, >=, || (OR), && (AND)

IF statements

```
if ( statement ) {          // if the statement is true, enter the block (inside {}).
                            // if false, skip to *.
}
*
```

IF/ELSE IF/ELSE statements

```
if ( statement) {           if statement is true, then enter the block and skip
                            the rest of the elseif/else blocks.
} else if ( statement ) {   if first statement is false, then try the next else if
                            statement.

} else {                    if all previous statements were false, enter the else block.

}
```

Question: What will be the outputs of the two codes below?

int n = 10;
if (n > 0)
  cout << "Hello ";
else if (n > 5)
  cout << "World";
	int n = 10;
	if (n > 0)
	  cout << "Hello ";
	if (n > 5)
	  cout << "World";

Output:	Output:

Question: What is the problem with this code snippet? What is the proper way to rewrite it?

```
int n = 5;
if (0 < n < 10)
        return true;
```

Question: Is this a valid C++ code? If Yes, what is the output?

```
int count = 3;
while (count-- > 0)
        cout << count << " ";
```

EXTRA: Look up **conditional operators.**

Ch 3: function basics: predefined functions, programmer defined functions, scope rules

You should be familiar with predefined functions:

- <cmath> sqrt
- <cstdlib> abs, rand, srand, exit

Writing Functions from Scratch

Function **declaration**:

```
<return_type> <function_name> (parameter (s) );        // used when working with multiple functions
```

Function **definition**:

```
<return_type> <function_name> ( parameter(s) ) {       function header looks the same as the function declaration,
        <function body>                                except it contains a function body instead of ending with a semi-colon
}
```

Example:

```
bool isEven (int n) {
        if (n % 2 == 0) return true;
        return false;
}
```

Question: Create a **round** function that takes in a double value and returns the nearest integer to that number.

Question: There are two issues with the following program. What are they?

```
#include <iostream>
using namespace std;

int main () {
        int p = 5;
        if (isEven(p)) {
                cout << “Even.” << endl;
        }
}

bool isEven (int n) {
        if (p % 2 == 0) return true;
        return false;
}
```

Ch 4: parameters and overloading: parameters (value vs reference), return types

You should have a mastery in the difference between pass-by-value and pass-by-reference.

Question: What is the output of the following program?

```
#include <iostream>
using namespace std;

int fun1 (int x, int y) {
        y = 1;
        x = 0;
        return y;
}

int fun2 (int &x, int &y) {
        y = 1;
        x = 0;
        return y;
}

int main() {
        int x = 10, y = 20;
        cout << fun1 (x, y) << endl;
        cout << x << " " << y << endl;
        cout << fun1 (y, x) << endl;
        cout << fun2 (y, x) << endl;
        cout << x << " " << y << endl;
}
```

Ch 5: arrays: using arrays, arrays as parameters, programming with arrays, multi-dimensional arrays

Question: What is the output of the following program?

```
#include <iostream>
using namespace std;

void fun (int &x) {
        x = 5;
}

int main() {
        int a[10];

        for (int i = 0; i < 10; i++) {
                i = i + 1;
        }

        cout << fun(a[0]);

        for (int i = 0; i < 10; i++) {
                cout << a[i];
        }


}
```