CS 162 Worksheet 1

1. C++ data type review: indicate if each the following matched with the correct type:

Constant	Type	Right/Wrong (correction)
`4.0`	int	
`5`	int	
`‘a’`	string	
`5.`	double	
`5`	char	
`“5.0”`	char	

2. Arithmetic Operators

Operator	Name	Example
`+`		
`-`		
`*`		
`/`		
`%`		

3. Relational operators: to perform comparison of variables, constants, or expressions in C/C++

Operators(s)	Meaning	Example
`==`		
`!=`		
`<`		
`>`		
`<=`		
`>=`		

4. Conditional Statements: if/else
   What will each implementation print if ‘grade’ stores 95?

Implementation 1:
```
if (grade >= 90) {
	cout << "A range" << endl;
}
else if (grade >= 80) {
	cout << "B range" << endl;
}
else if (grade >= 70) {
	cout << "C range" << endl;
}
else {
	cout << "Below C range!" << endl;
}
```

Implementation 2:
```
if (grade >= 90) {
	cout << "A range" << endl;
}
if (grade >= 80) {
	cout << "B range" << endl;
}
if (grade >= 70) {
	cout << "C range" << endl;
}
else {
	cout << "Below C range!" << endl;
}
```

What did you notice about if and else?
if:

else:

5. Logical Operators: to create compound conditions

Operators(s)	Meaning	Example
&&		
||		
!		

Quick check: Which of the following is NOT a condition to check if the integer x is in the range [-1 to 5]?

A. `x >= -1 && x <= 5`
B. `-1 <= x <= 5`
C. `!( x < -1 || x > 5)`
D. `x > -2 && x < 6`

6. Common mistakes

a. Using assignment operator (=) rather than equality check operator (==)
Correct the following code:

```
int x;
cin >> x;
if (x = 0)
      cout << “x is 0” << endl;
```

Tip: When comparing with a constant, many companies recommend flipping the order to:
`if (0 == x) { /*some code*/ }`
This way, the code won’t compile if you accidentally write:
`if (0 = x) { /*some code*/ }`

b. Using multiple if statements rather than if … else
Correct the following code:

```
int x, y;
cin >> x >> y; //takes two inputs, and store them into x and y, respectively
if (x != y)
      x = 5;
if (x == y)
      y = 7;
```

c. Wrong formulated conditions.
Correct the following code:

```
if (0 <= x <= 9) { /*some code*/ }

if (x == 0 || 1) { /*some code*/ }
```

7. Loops
   a. for loop: used when you DO know the number of times to iterate BEFORE the loop starts
      Ex: print out all multiples of 7 from 0 to 100, inclusive

   b. while loop: used when you DON'T know how many times to iterate before the loop starts
      Ex: let user guess my secret number until they are correct

```
int guess;
int secret_num = /* some code */;
cin >> guess;
// complete the rest….
```

      Tip: Use while loop whenever you see/use "until", until x == while not x
      For example: keep guessing until correct == keep guessing while not correct

   c. do-while loop: often used to run/play again. Loop body is executed at least once
      Ex: ask the user whether they want to run the program again, 1-yes, 0-no

   d. nested loop: The inner loop executes completely for each single iteration of the outer loop
      Ex: Trace through the execution of the following code and show what will be printed.

```
for (int i = 0; i < 2; i++) {                        i          j
      for (int j = 0; j < 3; j++) {
            cout << i << " " << j << endl;
      }
}
```