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	Wrong
`5`	int	Right
`'a'`	string	Wrong
`5.`	double	Right
`5`	char	Wrong
`"5.0"`	char	Wrong

2. Arithmetic Operators

Operator	Name	Example
+	Addition	x + y
-	Subtraction	y - x
*	Multiplication	x * 5
/	Division	y / 2
%	Modulus	y % 2

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

Operators(s)	Meaning	Example
==	equals	x == 2
!=	not equal to	x != 3
<	less than	1 < 4
>	greater than	y > 5
<=	less than or equals to	x <= y
>=	greater than or equal to	y >= x

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;
}
```

Prints:
A range

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;
}
```

Prints:
A range
B range
C range

What did you notice about if and else?

if: uses a conditional statement to selectively run code

else:

runs once the prior if statement returns false

5. Logical Operators: to create compound conditions

Operators(s)	Meaning	Example
&&	AND	if (x && y) {}
||	OR	if (y || z == True) {}
!	NOT	if (!x) {}

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;
```

Fixed:
if (x == 0) {}

      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;
else 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*/ }
```

Fixed:

(0 <= x && x <= 9)

(x == 0 || x == 1)

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

```
for (int i = 0; i < 100; i++) {
        if (i % 7 == 0)
{
                cout <<
i << endl;
}
```

   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….

while (guess != secret_num) {
        cout << "That's wrong, try again" <<
endl;
        guess_number();
}
cout << "That's correct!" << endl;
```

      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

```
play_again;
do {
        /* some code */
        cout << "Do you want to play again? 1 - yes, 0 -
no"
}
while (play_again);
```

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

i	j
0	0
0	1
0	2
1	0
1	1
1	2