Quest 8

200 EXP

CPSC121 SI

Craig: Welcome Back Programmer.

Craig: Last time we talked about if statements. This time we will be working with if statements on steroids.

Craig: We will look at nested if statements and logical operators.

Craig: Rex, please explain to our friend what a nested if statement is.

Rex: A nested if statement is an if statement inside of another if statement.

Craig: Exactly! We use this if we only want to check a condition if another condition is previously met.

Craig: For example! Say you want to check if a student is in CPSC 121 Section 7. However, before you check the section number, you need to make sure they are in the correct class. In addition, they must be a student at CSUF.

Craig: Here's an example of how to check this via nested if statements:

```
if(school == "CSUF")
{
   if(class == "CPSC")
   {
      if(section == 7)
      {
        cout << "You are in CPSC121-07 at CSUF\n";
      }
      else
      {
         cout << "Sorry, you are not in section 7.\n";
      }
   }
}
else
{
   cout << "Sorry, you are not a student of CSUF.\n";
}
```

Craig: As you can see, if the first condition is not met then the program does not need to check any other conditions. Also note how trailing else statements are linked with the most recent if statement. The first else statement is linked to if(section == 7) and the second else is linked to if(school == "CSUF").

The second else is linked to the school condition and not the class condition because the school condition is the most recently closed if statement.

Craig: nested if statements are effective but what if you want to just check everything at once?

Rex: You could use **Logical Operators**.

Craig: Correct! Three logical operators that you will find incredibly useful are AND, OR, and NOT.

Craig: The three operators are represented respectively by &&, ||, and !

Craig: Here are some examples of how you would use these:

```
int x = 6, y = 4, z = 4;

(x > y) && (y == z) //This is true because x is greater than y AND y is equal to z

(x > y) && (z > x) //This is false because x is greater than y, but z is not greater than x

(x == z) || (z == y) //This is true because at least one part of the condition is true

!(y > z) //This is true because y > z is false, but there is a NOT in front of it

!(x == z) && (y >= z) && (x > y) //This is true because all three parts are true
```

Craig: Another thing to be aware of with logical operators is that they follow a priority system. NOT and AND take priority over OR.

```
(2<3) || (6 > 8) && (0 > 3)
```

Craig: This expression evaluates the AND first, resulting in:

```
(2<3) || false
```

Craig: The OR will result in an output of true because 2 is less than 3.

Craig: Now what happens when you mix operators? There is a priority system for that too. Computers execute arithmetic operators, then relational operators, then logical operators.

```
8 < 2 + 7 || 5 == 6
```

Craig: In this expression, the 2 + 7 is evaluated first because it is an arithmetic operator. Then 8 < 2+7 is executed because it is the left most relational operator. Then 5 == 6 because relational operators before logical operators. Lastly, the OR is executed.

Craig: This quest is super simple. You just need to tell what the output of the following program is. Be careful though. The conditions and operators might get tricky

```
int x = 5, y = 6, z = 7;

if(x < y && y < z)
{
	cout << “Let the games begin.\n”;
	if( x – y >= z – y)
	{
		cout << “The games continue!\n”;
	}
	else
	{
		cout << “The game stopped?\n”;
	}
	x = 10;
	y += 4;
	z = x * y;
	if( !(y > z) || z < x + y )
	{
		cout << “This is easy.\n”;
	}
	else if ( x == y)
	{
		cout << “What just happened?\n”;
	}
	else
	{
		cout << “Can you make this any easier?\n”;
	}
}
else
{
	cout << “The games never started.\n”;
}
```

Craig: Easy EXP right? Show your SI leader what you got to make sure you traced that correctly.