If-else Statements

1. Download a basic “skeleton.html” from the FIT 100 calendar and rename the file. Place the javascript code below between the <body></body> tags.

```
var myProjectScore = 75;
document.writeln("My Project Score is: " + myProjectScore);
```

---

2. Replace the above with the following. Write down what the “alert box” says.

```
var myProjectScore = 75;
var neededPoints;

if (myProjectScore == 100) {
  perfectScore = true;
  neededPoints = 0;
} else {
  perfectScore = false;
  neededPoints = 100 - myProjectScore;
}
alert("Perfect Score? " + perfectScore + ", I got a " + myProjectScore);
```

---

3. Change the “myProjectScore” variable from 75 to 100. Write down what will the alert read.

4. Instead of assigning a static score (like 75 or 100) to the “**myProjectScore**” variable, we will write code so we can input our own score.

In the <head> of the document, within <script type="text/javascript"></script> tags, create a function called “**scoreCalculator**”

Within the body of your “scoreCalculator” function, add your code from part 2 above. This time however, instead of the static numerical value of 75, assign the “**myProjectScore**” variable the following statement.

```
document.getElementById("userInput").value;
```

Next, add the input box HTML code to the <body> of your HTML document.

```
Enter your Grade: <input id="userInput" type="text"
onchange="scoreCalculator()" />
```

Explain what each attribute of the “input” tag is doing.

- id
- type
- onchange (actually, “onchange” is an event handler)

your code should now look something like this

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
<title>Lab 7: Better Version</title>

<script type="text/javascript">

function scoreCalculator() {
var myProjectScore = document.getElementById("userInput").value;
var neededPoints;

if (myProjectScore == 100) {
  perfectScore = true;
  neededPoints = 0;
} else {
  perfectScore = false;
  neededPoints = 100 - myProjectScore;
}
alert("Perfect Score? " + perfectScore + ", I got a " + myProjectScore);
}

</script>
</head>

<body>
Enter a number: <input id="userInput" type="text"
onchange="scoreCalculator()"/>

</body>
</html>
```

A user can now input their score and check to see if it qualifies as a perfect score.

Relational operators are used to make comparisons between numerical values—to test the relationship between **two** numerical values.

variable1	<	variable2	Is variable1 **less than** variable2?
variable1	<=	variable2	Is variable1 **less than or equal to** variable2?
variable1	==	variable2	Is variable1 **equal to** variable2?
variable1	**!=**	variable2	Is variable1 **not equal to** variable2?
variable1	>=	variable2	Is variable1 **greater than or equal to** variable2?
variable2	>	variable2	Is variable1 **greater than** variable2?

When evaluating each of these statements, the outcome is a **Boolean value**, which is either **true** or **false.**

Look at the following piece of code from our previous example…

```
-------------------------------------------------
if (myProjectScore == 100) {
  perfectScore = true;
  neededPoints = 0;
} else {
  perfectScore = false;
  neededPoints = 100 - myProjectScore;
}
-----------------------------------------------------------------------
```

As of right now, when a user inputs his/her score, if and only if he/she inputs the number "100" will it be considered a perfect score. What if a user inputs 104 points? To you and I, 104 is a better than perfect score, but our "silly" program only knows a perfect score as 100, and nothing else.
**Very important hint for HW3. Algorithms don't know how to fill in missing steps or answers. We know that 104 is a perfect score, but our algorithm doesn't. Although computer programs may seem "smart," really they are only as smart as the individual(s) who programmed them. Algorithms and computers are useful because they can manage lots of details, and can do so without error (that is without errors from the programmer). In the example your just created, if you input a perfect score 1 million times, it will ALWAYS register as a perfect score. Do that with a human being, and there is a chance for a human error.

Now lets make our program smarter with **relational operators**. Alter your code so that a perfect score registers as anything greater than 100.

****To check your work here, enter a number greater than 100, say 110, and see what happens. If done correctly, your alert box should then say, "Perfect score? True, I got a 110.**

Lets make our program even "smarter" using **logical operators**. Logical operators allow you to compare the relationships between **more than just 2** numerical values.

Logical And
Ex: (age == 11 **&&** monthBorn ==12)
This translates to "age is equal to 11 AND monthBorn is equal to 12." In order for this to evaluate to true, someone would have to be both 11 years old, and born on December.

Logical Or
Ex: (age == 11 || monthBorn == 12)
This translates to "age is equal to 11 OR monthBorn is equal to 12," so anyone who is either 11 years old or born in December.

Logical Not
Ex: **!**(age ==11 || monthBorn == 12);
This translates to "age is not equal to 11 OR not equal to 12," so anyone who isn't 11 years old or born in December.

So in order to make our program smarter, we'd want a perfect score as someone who got over 100 and someone who did not cheat. We must now declare a new variable.

```
var didNotCheat = true;
```

Next you must alter the "if" conditional. Currently it looks like this.

```
if (myProjectScore == 100) {
```

Using a **logical operator**, alter the "if" condition above to satisfy the new requirements for a "perfect score." What did you add to your "if" conditional?

Next, if we change the didNotCheat variable to false like so

```
var didNotCheat = false;
```

What **logical operator** would we use to satisfy the requirements of a perfect score?

Basic Iterations-for loops

1. Download a **NEW** "skeleton.html" from my resources page, and place the following javascript code below between the <body></body> tags.

```
-------------------------------------------------------------------------------
var loopCount = 1;
for ( i=0; i<loopCount; i++ ) {
      document.writeln("Print this text. <br />");
}
-------------------------------------------------------------------------------
```

2. If you change the count from "1" and replace it with "5," what happens? Why does that happen?
3. If you would like to see the numerical value for "i" (it will help you see how many times the loop is running), replace the body of the "for loop" with the following:

   ```
   document.writeln("The iteration variable has the value " +  i  + "<br
   />");
   ```

4. You now should see the following results. Why does the value start at "0"?

The iteration variable has the value 0
The iteration variable has the value 1
The iteration variable has the value 2
The iteration variable has the value 3
The iteration variable has the value 4

5. Currently, the results displayed are showing that the loop has run a total of 5 times. Now change the loop so that it executes a total of 10 times.

`If done properly, your results should look like`

The iteration variable has the value 0
The iteration variable has the value 1
The iteration variable has the value 2
The iteration variable has the value 3
The iteration variable has the value 4
The iteration variable has the value 5
The iteration variable has the value 6
The iteration variable has the value 7
The iteration variable has the value 8
The iteration variable has the value 9

There were two ways you could have done this. By changing the assignment value of "loopCount" to 10, or by changing the portion of the "for loop" that checks for a limit condition (from "i<loopCount;" to "i<10").

6. Now adjust your "for loop" so it displays the following results.

The iteration variable has the value 0
The iteration variable has the value 2
The iteration variable has the value 4
The iteration variable has the value 6
The iteration variable has the value 8

7. Now adjust your iteration so it displays the following results.

The iteration variable has the value 10
The iteration variable has the value 9
The iteration variable has the value 8
The iteration variable has the value 7
The iteration variable has the value 6
The iteration variable has the value 5
The iteration variable has the value 4
The iteration variable has the value 3

The iteration variable has the value 2
The iteration variable has the value 1
The iteration variable has the value 0
The iteration variable has the value -1
The iteration variable has the value -2
The iteration variable has the value -3
The iteration variable has the value -4

8. Now adjust your iteration so it loops 5 times. It doesn't matter where you start, stop, or the amount you change the iteration variable, AS LONG AS IT ITERATES ONLY 5 TIMES.

   Download the image below and place it in the same directory as this lab. (http://www.cs.washington.edu/education/courses/cse100/CurrentQtr/lab07/lab7-2.jpg)

   Now use the "**statement**" below as the body of your iteration

   ```
   document.write("<img src='star.jpg' />");
   ```

   Your finished result should look like this. If it is not working right away, think about why the images are not displaying.