**CSCI 1300 - Intro to Computer Programming**
**Instructor: Fleming**
**Recitation 3**

This assignment is due **Friday, September 21st, by 11:55 pm**

- ***All components*** **(Cloud9 workspace and moodle quiz attempts) must be completed and submitted by Friday, September 21st at 11:55 pm for your solution to receive points.**
- **Recitation attendance is required to receive credit.**

---

**Objectives:**

- Understand what conditionals are and how to use them.

---

**Please follow the same submission guidelines outlined in Homework 3 description regarding Style, Comments and Test Cases. Here's a review below on what you need to submit for Recitation 3.**

**Develop in Cloud9:** For this recitation assignment, write and test your solution using Cloud9.
**Submission:** All three steps must be fully completed by the submission deadline for your homework to receive points. Partial submissions will not be graded.

1. ***Make sure your Cloud 9 workspace is shared with your TA:*** Your recitation TA will review your code by going to your Cloud9 workspace. *TAs will check the last version that was saved before the submission deadline*.
    - Create a directory called **Rec3** and place all your file(s) for this assignment in this directory.
    - Make sure to *save* the final version of your code (File > Save). Verify that this version displays correctly by going to File > File Version History.
    - The file(s) should have all of your functions, test cases for the functions in main function(s), and adhere to the style guide. Please read the **Test Cases** and **Style and Comments** sections included in the **Homework 3** write up for more details.
2. ***Submit to the Moodle Autograder:*** Head over to Moodle to the link **Rec 3**. You will find one programming quiz question for each problem in the assignment. Submit your solution for the first problem and press the Check button. You will see a report on how your solution passed the tests, and the resulting score for the first problem. You can modify your code and re-submit (press *Check* again) as many times as you need to, up until the assignment due date. Continue with the rest of the problems.

**CSCI 1300 - Intro to Computer Programming**
**Instructor: Fleming**
**Recitation 3**

---

**Conditionals:**

The *if* keyword is used to execute a statement or block, if, and only if, a condition is fulfilled.
Its syntax is:

```
if (condition)
{
    // Execute these statements if condition TRUE
}
```

Sometimes, when the condition in an if statement evaluates to false, it would be nice to execute some code instead of the code executed when the statement evaluates to true. Use else if to specify a new condition to test, if the first condition is false. The "else" statement effectively says that whatever code after it is executed if the previous if statements are FALSE.
Its syntax is:

```
if ( condition 1 )
{
    // Execute these statements if condition 1 is TRUE
}
else if ( condition 2 )
{
    // Execute these statements if condition 1 is FALSE and condition 2 is
TRUE
}
else
{
    // Execute these statements if FALSE
}
```

**REMEMBER: If an "if", "else if" or "else" condition is satisfied and entered, then the code will resume outside of the conditionals.*

**CSCI 1300 - Intro to Computer Programming**
**Instructor: Fleming**
**Recitation 3**

## Relational Operators

| | | |
|---|---|---|
| > | greater than | 5 > 4 is TRUE |
| < | less than | 4 < 5 is TRUE |
| >= | greater than or equal | 4 >= 4 is TRUE |
| <= | less than or equal | 3 <= 4 is TRUE |
| == | equal to | 5 == 5 is TRUE |
| != | not equal to | 5 != 4 is TRUE |

## Logical Operators

You can combine relational operators using *logical operators*

- **&&** (AND) returns true if and only if both operands are true
- **||** (OR) returns true if one or both operands are true
- **!** (NOT) returns true if operand is false and false if operand is true

## Some example code

```
1  #include <iostream>
2
3  using namespace std;
4
5  void numberRange(int number)
6  {
7      if(number > 0 && number <= 100)
8      {
9          cout << "This number is between 0 and 100!";
10     }
11     else if(number > 100 && number <= 200)
12     {
13         cout << "This number is between 101 and 200!";
14     }
15     else
16     {
17         cout << "This number is greater than 200!";
18     }
19 }
20
21 int main()
22 {
23     int number;
24     cout << "enter a positive number to see what number range it falls in!\n";
25     cin >> number;
26     numberRange(number);
27 }
28
```

**CSCI 1300 - Intro to Computer Programming**
**Instructor: Fleming**
**Recitation 3**

**Problems Set:**
Write a **function** for each of the following problems. You should first write your solution in Cloud9 and then copy and paste your function into the Moodle CodeRunner answer box.

## Problem 1

Write a function named **numberSign** that takes one integer parameter, returns nothing, and prints whether that integer is "negative", "positive", or "zero".

- Your function should be named **numberSign**
- Your function takes one input argument: an **integer** number
- Your function does not have a return value
- Your function prints/outputs to the console window a message, as specified above

Examples:

1. If the input argument is 4, the function prints “positive”
2. If the input argument is -4, the function prints “negative”
3. If the input argument is 0, the function prints “zero”

## Problem 2

Write a function called **calcPay** which takes two floating point parameters (type **double)** representing the number of hours worked and an hourly pay rate. The function returns the calculated total pay. Hours worked over 40 hours (overtime hours) should be calculated at 1.5x the pay rate (time and a half).

- Your function *MUST* be named **calcPay**.
- Your function should take **two** parameters *in this order*:
    - a floating point parameter for hours worked (type **double)**
    - a floating point parameter for pay rate (type **double)**
- Your function should return the total pay.

For example, if someone worked 41 hours and their pay rate is $10 their total pay is $415.

| | | |
|---|---|---|
| Regular Pay: | (40 hours * $10) | = $400 |
| Overtime Pay: | (1 hours * ($10 * 1.5)) | = $15 |
| Total Pay: | | = $415 |

**CSCI 1300 - Intro to Computer Programming**
**Instructor: Fleming**
**Recitation 3**

**Handling invalid input:**

What happens if the user accidentally calls **calcPay** with a pay rate of -$10 and 41 hours worked? We could calculate their pay using the formula, but the result would be -$415, which doesn't make much sense! So, if one or both the parameters (pay rate and hours worked) are negative, return -1.

## Problem 3

Write a function called **checkLeapYear** that takes a single integer argument representing a year and returns 1 if it is a leap year and 0 otherwise. This function should use a single if statement and Boolean operators.

- Your function *MUST* be named **checkLeapYear**.
- Your function should take **one** parameter: the year value, as a positive **integer**
- Your function should return 0 or 1, according to the leap year definition (see below).

What is a leap year? In general, years divisible by 4 are leap years. For dates after 1582, however, there is a *Gregorian correction*. Years divisible by 100 are not leap years but years divisible by 400 are. So, for instance, 1900 was not a leap year but 2000 was.