CS 5001 Intensive Foundation of CS
Homework 3: Boolean Functions

Due: 5:59am on Thursday, January 31st

The goal of this assignment is write predicate functions. A *predicate* is a function which returns `True` or `False`. You will also be asked to practice writing testing functions. **You should work individually on this assignment.**

1 Getting Started

You should start by creating a folder specifically for this assignment. Create a `README.txt` file including three things: your name, the course name, and the name of this assignment. You should also use it answer any questions that may be asked in this assignment along with including any comments that you want the grader to consider when grading your assignment. **You should not include any code in your `README.txt` file.**

2 Predicate Functions

Following the same process as we did in previous assignments, you will design, implement, and test the following functions. You should start by downloading the provided `2019spring-cs5001-homework03-StarterFiles.sip` archive provided on Bottlenose and extract the `predicateFunctions.py` and `predicateFunctionsTest.py` files into the folder that you created specifically for this assignment. You should modifying these files.

Testing predicate functions is different than the way we have been writing tests thus far in our assignments in two ways:

1. The number of possible outcomes for a predicate function is exactly 2 values: `True` or `False`. You should write tests for both possibilities.
2. Test cases usually forgo using the `expected` value. Instead it uses `assertTrue` and `assertFalse` . `assertTrue(expr, msg = None)` tests that `expr` is true, If false, test fails and print `msg`. Similarly, `assertFalse(expr, msg = None)` test that `expr` is false. If true, test fails and print `msg`.

2.1 Function: `xor`

In this first function, we will implement and test a function that implements *exclusive or* as it was defined in Lab 3. Your function should take two arguments representing the values of x and y in the definition, and return `True` or `False` appropriately. To thoroughly test this predicate function, you will want to test as many of the possible combinations from the *truth table*. *Hint:* to do this, use the boolean expression that was the solution to the last question from Lab 3.

2.2 Function: `isNegative`

Design, implement, and test a function called `isNegative` that takes a single numeric argument and returns `True` if the argument is *less than 0* and `False` otherwise. *Hint:* you can implement this using a relational operator.

2.3 Function: `isLeapYear`

Design, implement, and test a function called `isLeapYear` that takes a `year` as an argument and returns `True` if the year is a leap year and `False` otherwise. In order to determine whether the year is a leap year, write a boolean expression that evaluates the year using the following rules:

- year is a multiple of 4, but not a multiple of 100, or
- year is a multiple of 400

When testing this function, consider using the following: 2000, 2012, 2018, and 2100

Hint: to determine whether x is a multiple of y, use the *module* (remainder) operator. If x is a multiple of y, then when x is divided by y the remainder is 0.

3 Submit Files to Bottlenose

Before submitting, be sure that you have added a comment at the top of your source files that includes your name and the current date. Then create an archive of the folder that you created for this assignment containing all of the files that you have created and upload this archive to Bottlenose for grading. When you upload, be sure to check whether there are any auto-graders that evaluated your code and fix any issues pointed out by the auto-graders. Homework 3 Review will be available immediate after the deadline for this assignment has passed. **You will not be able to resubmit after completing the review**