CSC 106, Summer 2016, Homework #3
Due Friday, June 30 at 11:55pm
100 marks total, worth 6% of your final grade

Question 1: Programming Languages [30 marks]

The median of a set of numbers is the number which position is the middle position, when the numbers are sorted.
For example, the median of 2, 9 and 1 is 2; and the median of 2, 7 and 5 is 5.

Choose 3 languages from the list below and, for each language, write a program that reads three integer values from the user (X, Y and Z) and prints the median of the three. Some of the languages listed may be more challenging to use than others.

- Ada
- C or C++ (but not both)
- C#
- Fortran-95
- Java
- Common Lisp
- Pascal
- Perl
- Python

All of the languages above are supported by the online tools found in the IDE section at http://www.tutorialspoint.com/codingground.htm. If you develop your program using the tools on that website, you can download your source file by right-clicking on the file in the left-hand pane of the IDE, and selecting "Download file". A brief introduction to the tool will be given in the labs on July 17th.

Your submission for this question will consist of three files (one for each chosen language). Please name your files according to the table below.

Language	Filename
Ada	median3.adb
C	median3.c
C++	median3.cpp
C#	median3.cs
Fortran 95	median3.f95
Java	Median3.java
Lisp	median3.lisp
Pascal	median3.pas
Perl	median3.pl
Python 2.7	median3.py
Python 3.x	median3v3.py

Each program will be worth 10 marks. Programs that are not named correctly or do not run will automatically receive the mark of 0.

Bonus (optional): You will receive 10 bonus marks if you submit implementations for all 10 languages above. (As Bill Bird puts it: this is mostly a character building exercise, since the work required far outweighs the number of bonus marks offered)

Question 2: Debugging the PoisonedBottle [20 marks]

An important skill for a programmer is to be able to debug code, whether hers/his, or someone else's. For a little practice, you are given an implementation of the Poisoned Bottle algorithm in Python 2.7 (won't work with 3.x).

Who ever wrote that code followed the pseudocode line by line, however, this implementation does not work, as the output does not fit the input. I was testing the code with deathBin = [0,0,1,1], expecting 3 to be the output, but it returned 12.

I am sending the code to you and ask you to fix the bug. As you learned in your labs, you can run from your terminal using the command:

```
python poisonedBottle.py
```

The bug is in the isPoisoned(deathBin) definition. You can modify the body of the function to fix the bug, and the main to test with different input, but
do not change the lines starting with **def** or **if __name__**,
or we won't be able to test our code.

Submit your fixed code on the connex page. Do not modify the name of the file (poisonedBottle.py).

Question 3: Quick Sort Efficiency [20 marks]

a) (4 marks) Our quicksort pseudocode chooses the very leftmost element as the pivot. How many calls to quicksort (including the first call) will there be on the following array? [1,2,3,4,5,6]

b) (4 marks) How many calls to quicksort (including the first call) for this array? [3,1,4,5,6,2]

c) (6 marks) Why are there so many more calls on the first array, even though it has exactly the same elements (just a different order)?

d) (6 marks) Some implementations of quicksort choose a pivot at random from the array (rather than always choosing the leftmost element). Why does that alleviate the problem?

Question 4: Quick Reverse Sort [30 marks]

a) (15 marks) Change quicksort_reverse.py so that it sorts the array in reverse order. (Do not just print it backwards!)

b) (15 marks) Change the reversed quicksort_reverse.py so that it chooses the pivot randomly instead of just selecting the leftmost element.
i.e. choose a random element in the array to be the pivot, rather than always choosing A[low].

You can generate random numbers using random.randint(low, high)
See https://docs.python.org/2/library/random.html

You can run this code with the command:
python quicksort_reverse.py

There are a number of print lines that have been commented out. They start with #print …
To execute theses lines, you need to uncomment then… which means delete the # at the beginning of the line.

Submit your modified code without changing the name of the file (quicksort_reverse.py).
As for the previous questions, you can modify the body of the functions, but **do not change** the lines starting with **def** or **if __name__**,
or we won't be able to test our code.