Due on January 28, 2019
64 points total

General directions: We will exclusively use Python 3 for our programming assignments, and allow only the use of modules in the Python 3 standard library unless explicitly specified otherwise on an individual assignment basis. This forbids the use of common third-party libraries such as Numpy, Sympy, etc., but not the use of math or io.

Unless specified otherwise, for the X-th homework, download the single "hwX_skeleton.py" file from the course website, and rename it to "hwX.py" on your machine. When you are done and ready to submit, upload your file named **exactly** "hwX.py" on Gradescope for assignment "HW X (Programming)." When you upload your file, the autograder will run a simple test for each function so that you can confirm it was properly uploaded and executed. Generally, if an assignment involves printing or writing a file in a specific format, there will be at least one simple test that checks your output is formatted as we expect. These tests are not worth any credit — once the due date is over, your submission will be graded by a collection of additional test cases.

All answers to non-programming questions must be typed, preferably using LaTeX. If you are unfamiliar with LaTeX, you are strongly encouraged to learn it. However, answers typed in other text processing software and properly converted to a PDF file will also be accepted. To submit your file, upload your PDF on Gradescope for assignment "HW X (PDF)." Handwritten answers or PDF files that cannot be opened will not be graded and will not receive any credit.

Finally, please read the detailed collaboration policy given on the course website. You are **not** allowed to discuss homework problems in groups of more than 3 students. **Failure to adhere to these guidelines will be promptly reported to the relevant authority without exception.**

Point values: Every problem has a specified amount of points which are awarded for the correctness of your solutions. In addition, each proof-oriented problem has an additional **style point**. In the homework handout, this is signified by a "+1" in the point value. To earn this point, your solutions should be clear, well organized, and easy to follow. This is to encourage not only perfectly correct solutions, but well presented ones.

Problem 1 (21 points)
One way to test whether an integer n is prime is to check if n is divisible by any positive integer i where $i = 2, i = 3, \ldots, i = \lfloor\sqrt{n}\rfloor$. (Note: The 'floor' of a number x, denoted $\lfloor x \rfloor$, is the greatest integer that is at most x). If no such i divides n, then n must be prime, otherwise it is not prime; you will prove this fact as part of the following.

(a) **(10+1 points)** Prove the last statement above; that is, prove that any positive integer n that is not prime has a factor (other than 1) less than or equal to $\sqrt{n}$.

(b) **(10 points)** Implement the procedure above as the function `isPrime(x)` in hw1.py. The input to the function is a positive integer x, and the output is TRUE if x is prime, and FALSE otherwise.

Problem 2 (32 points)
Let n be a positive integer with k digits. Consider the following procedure. "Split" n into two numbers by taking the leftmost $k - 1$ digits and the rightmost digit, subtract twice the latter from the former, and take the absolute value of that difference. For example, performing this procedure on 1234 yields 115. It turns out that the number obtained by this procedure is divisible by 7 if and only if the original number is divisible by 7; you will prove this fact as part of the following.

(a) **(10+1 points)** Prove that a number n is divisible by 7 if and only if the result from performing the procedure above on n is divisible by 7.

(b) **(10+1 points)** The previous part implies that procedure above can be repeatedly successively, always maintaining that the resulting number is divisible by 7 if and only if the original number is as well. Show that for any positive integer x with at least two digits, repeating the procedure above will eventually result in a number with one digit. For example, starting with 1234, applying the procedure twice yields 1.

[Hint: Show that for any positive integer x with at least two digits, the number obtained by the procedure above always results in a number strictly less than x.]

(c) **(10 points)** Using the two facts above, implement the procedure above as a *recursive* function `isDivBySeven(x)` in hw1.py as follows. For any input non-negative integer x, `isDivBySeven(x)` writes x to STDOUT (which is already implemented for you), and then:

1. If x is a single digit (less than 10), TRUE (the Boolean type) is returned if x is divisible by 7, and FALSE is returned otherwise.
2. Otherwise, the procedure above is applied to x to obtain a new number y, and then `isDivBySeven(y)` is returned.

For example, by calling `isDivBySeven(1234)`, the numbers 1234, 115, and 1 will be printed to the console (each number on its own line), then FALSE will be returned.

Problem 3 (10+1 points)
Suppose a, b, and n are positive integers such that $b > a$. Prove the following:

If $b - a$ is even then $a^n + b^n$ is even.