02-201 / 02-601 Homework 9:
Python & the Goldbach conjecture

Due: Dec. 1 at 11:59pm

1. Set up

If you have a Mac:

- Open a Terminal, and type `python`. This should start an interactive Python session and ensure you can run Python.

If you have Windows:

- Download and install Python **2.7.8** (or any Python 2 version ≥ 2.6). This is **not** the latest version: Python 3 is incompatible with Python 2 in a few ways. Python 2 and 3 are not that different, but most code and libraries still use Python 2, so that's what we'll use for this assignment. You can find it here:

  `https://www.python.org/downloads/release/python-278/`

- Open a terminal and run Python with the command `C:\DIR\python` where `DIR` is the location at which you installed Python (often something like `C:\Python278`).

2. Assignment

2.1 The Goldbach conjecture

Perhaps the most famous open question in number theory is: Can every even number > 2 be written as the sum of 2 primes? No one can prove this, and no one can find a counter example. People have been trying to prove this for 272 years.

You'll write a few Python functions to play around with this conjecture.

2.2 Python Tutorial

Read sections 1–5 of the tutorial here:

`https://docs.python.org/2/tutorial/index.html`

2.3 What you should do

Create a text file called `goldbach.py` and write the following functions in it:

1. Write a function `primes(n)` that returns a sorted list of the prime numbers $\leq n$ (2 is the first prime number). Use the "sieve of Eratosthenes" algorithm from the slides for lecture 7.

   For example:

```
>>>  print primes(13)
[2, 3, 5, 7, 11, 13]
```

2. Write a function `sumOfPrimes(k)` that returns two primes a and b with $a \leq b$ such that $a + b = k$ or returns $0, 0$ if no such primes exist.

   For example:

   ```
   >>> print sumOfPrimes(10)
   (3, 7)
   ```

3. Write a function `allSumOfPrimes(k)` that returns a list of *all* pairs (a, b) such that a and b are prime, $a \leq b$, and $a + b = k$. This is similar to `sumOfPrimes` except instead of a single pair $a + b = k$, you return a list of all the ways k can be written as the sum of two primes. If there are no (a, b) pairs, return the empty list `[]`.

   For example:

   ```
   >>> print allSumOfPrimes(48)
   [(5, 43), (7, 41), (11, 37), (17, 31), (19, 29)]
   ```

4. Write a function `goldbach(k)` that tests all the even integers $\leq k$ to see whether they can be written as the sum of 2 primes. It should return a list of triples (z, a, b) such that all these conditions hold:

   - $z = a + b$,
   - a and b are prime, $a \leq b$,
   - z is even, > 2, and $\leq k$,

   for every integer z for which such a pair a, b exists. The list should be sorted by increasing z. Your function should also return a boolean value that is `True` if every even integer $\leq k$ is represented within the list, or `False` otherwise.

   For example:

   ```
   >>> print goldbach(10)
   ([(4, 2, 2), (6, 3, 3), (8, 3, 5), (10, 5, 5)], True)
   ```

5. Write a function `goldbachWidth(k)` that returns a dictionary (map) `D` such that `D[z]` is the number of ways each even number $2 < z \leq k$ can be written as the sum of two primes.

   For example:

   ```
   >>> print goldbachWidth(25)
   {4: 1, 6: 1, 8: 1, 10: 2, 12: 1, 14: 2, 16: 2, 18: 2, 20: 2, 22: 3, 24: 3}
   ```

   since, for example, $14 = 7 + 7$ and $14 = 3 + 11$.

2.4 Tips

You can create any auxiliary functions you want to help you write the above functions.

To test your code, you should create a `main()` function that calls these functions. Unlike in Go, you have to explictly call `main()` in your Python programs. For example, at the bottom of your file you can write:

```
def main():
    # put your test code here
    print primes(13)
    print sumOfPrimes(10)
    print allSumOfPrimes(48)
    print goldbach(10)
    print goldbachWidth(25)

if __name__ == "__main__": main() # call main if program run from command line
```

You can run your program via:

```
python goldbach.py
```

When you run your program this way, Python automatically sets a variable called `__name__` to the string `"__main__"` so you can call main if you want to. The autograder won't use your `main()` function, so you can put whatever code you want there to use while you are testing your functions.

2.5 Tips on how to start

Write and test the `primes(k)` function first. Be sure that if k is a prime number your output includes it.

Then write the remaining functions, testing each one as you go.

3. Learning outcomes

After completing this homework, you should

- have experience writing Python code
- practiced picking up a new language
- be familiar with the Goldbach conjecture