Homework 1:
Imperator Furiosharp

Due February 3 by the start of lecture.

Complete both of the programs below in F#.

Program 1: Someone Set Up Us the Bomb

Program a simple text-based game in which the player attempts to shoot cannonballs at a target that is some distance away. The player repeatedly enters an angle to fire the cannon at and an amount of gunpowder to use. The game calculates how far the cannonball will fly under those parameters and continues until the cannonball falls within 1 meter of the target. Program the game in this way:

1. A function `placeTarget`, which takes no parameters and returns a **random** float value between 0 and 1000 meters, inclusive. Review how to declare a function that takes no parameters. Use `(new Random()).NextDouble()` to generate a random float from 0.0 to 1.0. The return value represents the distance at which the target is placed at the start of the game.
2. A function `getAngle`, which takes no parameters and prompts the user to enter an angle to fire the cannon at. An angle must be between 0 and 90 degrees. Read a line of text from the console and convert it to a float. Repeatedly ask for an angle (in a loop) until a valid angle is entered. Return the angle.
3. A function `getGunpowder`, which takes no parameters and prompts the user to enter a **positive** float for the amount of gunpowder to use when firing the cannon. Return the amount.
4. A function `calculateDistance`, which takes an angle and an amount of gunpowder, and returns the total horizontal distance that a projectile fired at the given angle will fly. The angle is in degrees; the gunpowder amount is in kilograms, and each kilogram of gunpowder results in 30 meters/second of initial velocity. You will find `Math.Sin` and `Math.Cos` useful, but note that they operate on **radians**, not degrees. Use 9.81 m/s^2 for acceleration due to gravity.
5. A function `isHit`, which takes the location of the target and the distance that the cannonball fires, and returns true if and only if the cannonball falls within 1.0 meter of the target. **Do not** use any `let` statements in this function.
6. A `main` function that plays the game:
   (a) Generate a target distance. Print the distance to the user.
   (b) Get the angle and the gunpowder.
   (c) Calculate the distance the ball travels.
   (d) If it's a hit, print a message and terminate. Otherwise, print how far away from the target the cannonball landed (indicate "short" or "long" if the ball did not travel far enough / traveled too far, respectively), and repeat steps (b) through (d).
   (e) Note: F# does not have a **break** statement or a way to **return** early from a function.

Program 2: Project Euler

Solve Problem 10 on ProjectEuler.net using the F# programming language. Do not look up an answer and copy it. You **must** solve the problem by structuring your code this way:

1. A function `isPrime`, which takes a single argument `n` and returns `true` if the argument is a prime number. (Hint: starting with $i = 2$, determine if i is a factor of `n`. Check the next value of i if it is not. Stop when you reach the largest value of i that could be a factor of `n`, recognizing that all of `n`'s

factors j have another integer j^* that forms a *factor pair* with j, so that if you have already checked j to see if it is a factor of `n`, you **don't** have to also check $j*$, particularly if it is larger than j...)

2. A function `sumPrimes`, which takes a single argument `max` and returns the sum of all prime numbers up to `max`, inclusive. Use **imperative-style** programming for this function: declare a `sum` variable, iterate a `while` loop from `3` up to `max` increasing by `____` (fill in the blank), check to see if each value in the iteration `isPrime`, and if it is, mutate the `sum`. Don't forget to account for the number `2` being prime. Break when you reach `max`.
3. A main function that solves the problem.

Deliverables

Turn in the following when the lab is due:

1. Your source code file(s).
2. The output of program #1, showing the user win the game.
3. The output of program #2, consisting of the calculated summation.