F# Tutorial

Pipe-Forward Operator

February 23, 2018

1 Modified Project Euler Solutions

`IsPrime` Function Provided

The following function determines whether a positive integer `x` is a prime number or not. It is already provided, and we do not need to re-implment it.

```
let IsPrime x =
    let squareRoot = x |> double |> sqrt |> int
    if x = 1 then false
    else if x = 2 then true
    else if x % 2 = 0 then false
    else
        [3 .. 2 .. squareRoot]
        |> List.forall (fun i -> x%i <> 0)

// val IsPrime: x:int -> bool
```

Question 1

`https://projecteuler.net/problem=1`

Original Question. *Implement a function that sums up all multiples of* 3 *or* 5 *in a list.*

```
let SumMultiplesOf3Or5 xList =
    xList
    |> List.filter (fun x -> x % 3 = 0 || x % 5 = 0)
    |> List.sum
```

Question 2

Original Question. *The Fibonacci sequence (starting with 1 and 2) looks something like:*

$$1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

(For example, $1 + 2 = 3, 2 + 3 = 5, 3 + 5 = 8$, etc.)

Find the sum of all even-valued fibonacci numbers below 4 million.

1. We will first test whether the 41st fibonacci number exceeds four million or not.

```
let first40FibNumbers =
    [1 .. 40]
    |> List.scan (fun (x,y) _ -> (y, x + y)) (1,2)
// Result: [(1,2); (2,3); ......; (267914296, 433494437)]
```

   And so, we only need to consider the first 40 or 41 Fibonacci number. (In fact, we do not even need to consider beyond the 32th number)

2. Sum all even-valued fibonacci numbers below 4 million.

```
let fibSum =
    [1 .. 40]
    |> List.scan (fun (x,y) _ -> (y, x + y)) (1,2)
    |> List.map (fun (x,y) -> x)
    |> List.filter (fun x -> x % 2 = 0)
    |> List.filter (fun x -> x < 4000000)
    |> List.sum
// Result: 4613732
```

Question 3

Exercise (Euler Project Question 3)

`https://projecteuler.net/problem=3`

Please see the next section, where we approach the original Question 3.

Modified Question. *Write a function that takes a list of (positive) integers, and returns the largest prime number in that list.*

```
let FindLargestPrime intList =
    intList
    |> List.filter (IsPrime)
    |> List.max
```

Question 4

https://projecteuler.net/problem=4

Original Question. *A palindromic number reads the same from left-to-right or right-to-left.*

The largest palindromic number made from the product of two 2-digit numbers is $9009 = 91 \times 99$.

Find the largest palindrome made from the product of two 3-digit numbers.

The following `IsPalindrome` function that is already implemented. You do not need to re-implement it.

```
let ReverseString (xString: string) =
    new string (xString.ToCharArray() |> Array.rev)

let IsPalindrome xString =
    (ReverseString xString) = xString

let palindromeResult1 = IsPalindrome "ASDF"
// false

let palindromeResult2 = IsPalindrome "ABCCBA"
// true
```

Find the largest palindrome number which is a product of two 3-digit numbers $a \times b$, where $100 \leq a \leq 999$, and $100 \leq b \leq 999$

```
let findProductPalindrome =
    List.allPairs [100 .. 999] [100 .. 999]
    |> List.map (fun (a,b) -> a * b)
    |> List.filter (fun product ->
        product
        |> string
        |> IsPalindrome
    )
```

Question 5

`https://projecteuler.net/problem=5`

Modified Question. *Given a list of integers, find the lowest common multiple (LCM) of all those numbers. (Assume no integer overflow)*

Remark: We are given the following GCD and LCM functions. We do not need to re-implement them.

```
let rec gcd x y =
    if x < 0 || y < 0 then failwith "cannot accept negative numbers"
    if x > y then gcd y x
    else if x = 0 then y
    else gcd (y % x) x

let lcm a b =
    a * b / (gcd a b)
```

Solution:

```
let lcmOfList xList =
    xList
    |> List.fold lcm 1

let result11 = lcmOfList [1 .. 10]
// Result: 2520

let result12 = lcmOfList [2;3;4;6;8;12]
// Result: 24
```

Remark: This method will FAIL for `xList = [1 .. 20]` because of integer overflow. Please see the next section on how to handle the original question.

Question 6

`https://projecteuler.net/problem=6`

Original Question. *Given a list of integers $x_1, x_2, \ldots, x_n$, write a function that calculates the following:*

$$\left(\sum_{i=1}^{n} x_i\right)^2 - \left(\sum_{i=1}^{n} {x_i}^2\right)$$

```
let ProjectEulerProblem6 xList =
    let sumOfSquares =
        xList
        |> List.map (fun x -> x * x)
        |> List.sum

    let sum =
        xList
        |> List.sum

    // return
    (sum * sum) - sumOfSquares
```

```
let result13 = ProjectEulerProblem6 [1 .. 100]
```

Question 7

`https://projecteuler.net/problem=7`

Original Question. *The list of prime numbers are* $2, 3, 5, 7, 11, 13, \ldots$. *We can see that the* 6*th prime number is* 13.

What is the 10001*th prime number?*

We start with a random guess of 500000:

1. **Solution part (a)**: How many prime numbers are there between 2 and 500000?

```
let numberOfPrimesWithinRange =
    [2 .. 500000]
    |> List.filter (IsPrime)
    |> List.length
```

   Expected answer: 41538.

2. **Solution part (b)**: What is the 10001th prime number between 2 and 500000?

   Because of 0-based indexing, we use `(List.item 10000)` to find the 10001th prime number (which is between 2 to 500000).

```
let find10001thPrime =
    [2 .. 500000]
    |> List.filter (IsPrime)
    |> List.item 10000
```

Question 8

https://projecteuler.net/problem=8

Modified Question. *Given a list of digits, find four adjacent digits with the largest product. For example, in the following number:*

7316717653133062491922511**9674**4265747423553491949 34

The 4 consecutive digits that gives the largest product is $9 \times 6 \times 7 \times 4 = 9674$
(Notice that this line is the first line in the original question)

```
let digitList =
    [7;3;1;6;7;1;7;6;5;3;1;3;3;0;6;2;4;9;1;......]

let result8 =
    digitList
    |> List.windowed 4
    |> List.map (fun x -> x, ListProduct x)
    |> List.maxBy (fun (_,product) -> product)
// val result8 : int list * int = ([9;6;7;4], 1512)
```

Please see the next section on how we approach the original question.

Question 9

https://projecteuler.net/problem=9

Original Question. *Find the only Pythagorean triplet a, b, c that satisfy:*

$$a < b < c, \qquad a + b + c = 1000, \qquad a^2 + b^2 = c^2$$

```
let FindPythagoreanTriple =
    List.allPairs [1 .. 1000] [1 .. 1000]
    |> List.filter (fun (a,b) ->
        let c = 1000 - a - b
        a * a + b * b = c * c
    )
// Result: [(200, 375); (375, 200)]
```

Question 10

Exercise (Euler Project Question 10)

https://projecteuler.net/problem=10

Please see the next section where we work with large numbers (and potential integer overflow)

Modified Question. *Given a number $N < 200,000$, find the sum of all prime numbers between 2 and N.*

```
let TotalSumOfPrimeLessThan N =
    [2 .. N]
    |> List.filter (IsPrime)
    |> List.sum
```

2 Original Project Euler Solutions

Question 1

We did not modify Question 1.

Question 2

We did not modify Question 2.

Question 3

`https://projecteuler.net/problem=3`

Question. *Given an integer Z, write a function that finds the largest prime factor of Z. e.g. The prime factors of* 13195 *are* $5, 7, 13, 29$*, and so the largest for* 13195 *is* 29*.*

Problem Analysis

Remark: Given an integer Z, it is possible that the largest prime factor of Z is greater than $\sqrt{Z}$

- Example: $3 \times 7 = 21$. The largest prime factor is $7 > \sqrt{21} \approx 4.58$.
- Example: $6 \times 11 = 66$. The largest prime factor is $11 > \sqrt{66} \approx 8.12$.

To solve this question, we need some additional mathematical consideration (which is not quite directly related to programming).

1. Let $S_1 = \{a_1, \ldots, a_n\}$ be all the factors of Z (not necessarily prime factors) between 1 and $\sqrt{Z}$. This set will always contain at least one element: $a_1 = 1$.
2. Let $S_2 = \left\{\dfrac{Z}{a_1}, \ldots, \dfrac{Z}{a_n}\right\}$. These are all the factors of Z between $\sqrt{Z}$ and Z. This set will always contain at least one element: $\dfrac{Z}{a_1} = Z$.
3. So, $S_1 \cup S_2 = \left\{a_1, \ldots, a_n, \dfrac{Z}{a_1}, \ldots, \dfrac{Z}{a_n}\right\}$ are all the factor of Z (not necessarily prime factors).
4. Out of our list of candidates $S_1 \cup S_2$, which number is the $\underline{\text{largest}}$, $\underline{\text{prime}}$ number?

Working with `BigInteger`:

We will need an `IsPrimeBigInteger` function that helps us check whether a `BigInteger` is a prime number or not.

```
let IsPrimeBigInteger x =
    let squareRoot = x |> double |> sqrt |> BigInteger
    if x = BigInteger(1) then false
    else if x = BigInteger(2) then true
    else if x % BigInteger(2) = BigInteger(0) then false
    else
        [BigInteger(3) .. BigInteger(2) .. squareRoot]
        |> List.forall (fun i -> x%i <> BigInteger(0))
```

Code Solution

```
open System.Numerics

let FindLargestPrimeFactor (Z: BigInteger) =
    let approxSqrt = Z |> double |> sqrt |> BigInteger

    // Find factors of Z between [1 .. sqrt(Z)]
    // Not necessarily prime factors.
    let list1 =
        [BigInteger(1) .. approxSqrt]
        |> List.filter (fun x -> Z % x = BigInteger(0))

    // Find factors of Z between [sqrt(Z) .. Z]
    let list2 =
        list1
        |> List.map (fun a -> Z / a)

    // List.append combines the two lists.
    let combinedList =
        List.append list1 list2

    combinedList
    |> List.filter (IsPrimeBigInteger)
    |> List.max
```

Question 4

We did not modify Question 4.

Question 5

`https://projecteuler.net/problem=5`

Original Question. *Find the least common multiple (LCM) of* 1 *to* 20*.*

```
open System.Numerics

let rec GCD x y =
    let zero = BigInteger(0)
    if x < zero || y < zero then failwith "CANNOT ACCEPT
  NEGATIVE NUMBERS"
    if x > y then GCD y x
    else if x = zero then y
    else GCD (y % x) x

let LCM x y =
    x * y / (GCD x y)

let result =
    [1 .. 20]
    |> List.map (BigInteger)
    |> List.reduce LCM
```

Question 6

We did not modify Question 6.

Question 7

We did not modify Question 7.

Question 8

`https://projecteuler.net/problem=8`

Original Question. *In the webpage, a* $1000 - digit$ *number is provided.*

The four adjacent digits in the 1000*-digit number that have the greatest product are* $9 \times 9 \times 8 \times 9 = 5832$.

$$\mathit{7316717653133}......\mathit{31998900}.......\mathit{2963450}$$

Find the thirteen adjacent digits in the 1000*-digit number that have the greatest product.*

```fsharp
open System.Numerics

let bigIntegerString =
    "7316717653133062491...........20752963450"
// Please actually fill in the actual digits in the actual F# file!

let result =
    bigIntegerString
    |> Seq.map (string)
    |> Seq.map (int)
    |> Seq.map (BigInteger)
    |> Seq.windowed 13
    |> Seq.map (Seq.reduce (*))
    |> Seq.max
```

Remark: You can consider `bigIntegerString` as one long string, or a character `char` array.

So, the first `Seq.map (string)` converts an array of `char` to an array of `string` (where each string has only one letter/digit).

Question 9

We did not modify Question 9.

Question 10

`https://projecteuler.net/problem=10`

Original Question. *The sum of the primes below* 10 *is* $2 + 3 + 5 + 7 = 17$

Find the sum of all the primes below two million $(2,000,000)$.

```fsharp
open System.Numerics

let Version2_TotalSumOfPrimeLessThan N =
    [2 .. N]
    |> List.filter (IsPrime)
    |> List.map (BigInteger)
    |> List.sum

// Remark: The code below can take 10 seconds, as this is not the most optimal algorithm.
let result17 = Version2_TotalSumOfPrimeLessThan 2000000
// Result: 142913828922
```