Project Euler

Sean Go

January 2018

1 Problem 1

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000.

1.1 Brute force or Comprehension

Haskell

```
# dough
sum [n | n <- [1..1000-1], n `mod` 5 == 0 || n `mod` 3 == 0]
# lazcatluc
sum [3,6..999] + sum [5,10..999] - sum [15,30..999]
```

Listing 1: P1. Haskell Comprehension

Python

```
# johanlindberg
sum([x for x in range(1000) if x % 3== 0 or x % 5== 0])
```

Listing 2: P1. Python Comperhension

Assembly

```
        ; for each integer from 1 to 1000
        mov ecx, 3

        mov esi, 3
        mov edi, 5

        xor ebx, ebx        ; sum

_0:        mov eax, ecx
        xor edx, edx
        div esi
        test edx, edx
        je _yes

        mov eax, ecx
        xor edx, edx
        div edi
        test edx, edx
        jne _no

_yes:        add ebx, ecx

_no:        inc ecx
        cmp ecx, 1000
        jne _0
```

Listing 3: P1. Assembler Brute Force

1.2 Math

The sum of consecutive integers from 1 to c is

$$\sum_{k=1}^{c} k = \frac{c(c+1)}{2}$$

c , in this case, represents the count of numbers under N divisible by d. This is

$$c = \left\lfloor \frac{N-1}{d} \right\rfloor$$

The sum of all $3n$ is $\frac{3t(t+1)}{2}$, wehre $t = \left\lfloor \frac{1000-1}{3} \right\rfloor$;

The sum of all $5n$ is $\frac{5t(t+1)}{2}$, wehre $t = \left\lfloor \frac{1000-1}{5} \right\rfloor$;

The sum of all $15n$ is $\frac{15t(t+1)}{2}$, wehre $t = \left\lfloor \frac{1000-1}{15} \right\rfloor$;

The code:

```
def PE1(N):
    t=(N-1)//3; f=(N-1)//5; x=(N-1)//15
    return 3*t*(t+1)/2 + 5*f*(f+1)/2 - 15*x*(x+1)/2

print PE1(1000)
```

Listing 4: P1. Python Math

2 Problem 2

Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:
$1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...$
By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.

2.1 Brute Force

Assember. Only x86 has an easy way to swap variables (XADD) - other langauges require a temp varaible or XOR trick. Only x86 has an easy way to test for even numbers (bit 0 is zero) - other language use Mod() function.

```
        mov ecx, 1
        mov edx, 0
        xor ebx, ebx        ; sum
_0:        test ecx, 1
        jne _odd
_even:        add ebx, ecx
_odd:        xadd ecx, edx
        cmp ecx, 1000000
        jc _0
```

Listing 5: P2. Assembly. bitRAKE

List of source codes