MOISE GABRIEL

IMPERATIVE PROGRAMMING 2016

QUESTION 2

```
object Composite{
    // (a)
    def record (f: (int => int), emit: (int => Unit)) ={
        var mmax = 0
        var i = 1
        // Invariant i : we have called emit(j) for each j in [1..i) such that f(j) > f(k)
for all k∈ [1..j)
        while (1)
        { var x = f(i)  // f is called once for each value of i = 1,2,...
          if (x > mmax)
          { emit(i)
            mmax = x
          }
          i += 1
        }
    }

    // (b)
    def divisors (n : int) : int = {
        var div = 0
        var i = 1
        // invariant i : div is the number of divisors of n in [1..i)
        while (i <= n / 2)
        { if (n % i == 0) div += 1
          i += 1
        }
        // for every n/2 < i < n , i cannot divide n
        div += 1  // n divides n
        return div
    }
```

(c) The function divisors(n) takes O(n) time, therefore record(divisors, print) requires $O(n^2)$ time until it reaches the print(n) procedure, for any highly composite number n.

(d) $n = p_1^{i_1} \cdot p_2^{i_2} \cdot \ldots \cdot p_k^{i_k}$

Every divisor of n is then of the form $p_1^{a_1} \cdot p_2^{a_2} \cdot \ldots \cdot p_k^{a_k}$, with $0 \le a_j \le i_j$, for $j \in \{1, 2, \ldots, k\}$. By the product rule, there are $(i_1+1)(i_2+1) \cdot \ldots \cdot (i_k+1)$ distinct divisors for n (distinct because they have different prime factorisations).

```
// (e)
def divisorsFast (n: int) : int = {
    var aux = n
    var div = 1
    var i = 2
    // Invariant I: div = (i1+1)(i2+1)·...·(it+1), where i1,i2,...,it are the exponents of p1,p2,...,pt
with p1 < p2 < ... < pt < i are all the prime divisors of n up to (but not including) i
    while (aux != 1)
    {
      var ndiv = 0
      while (aux % i == 0) {ndiv += 1; aux = aux / i}
      div = div * (ndiv+1)
      if (i == 2) i += 1
      else i += 2 //skipping the even numbers
    }
    return div
}
```

(f) The "divisors" function needs n div and mod operations for every n, and the "divisorsFast" function needs $2 * \text{div}(n)$, where $\text{div}(n)$ = the number of divisors of n, so I would definitely expect the second function to be faster since $\text{div}(n) < n$.