IT UNIVERSITY BFNP SPRING 2015

Assignment 7 for Thursday 12 March and Thursday 19 March 2015
Version 1.1 of 2015-03-14 sestoft@itu.dk

These exercises concern parallel programming in F#.

You should build on the lecture's example code, found in file
http://www.itu.dk/people/sestoft/bachelor/bfnp20150312.zip

Exercise 1. Run the slow Fibonacci computations from the lecture's examples on your own computer. Use the #time directive to turn on timing, and see what is the best speed-up you can obtain for computing, say, the first 43 Fibonacci numbers using Async.Parallel. This may be quite different on MS .NET than on Mono.

Exercise 2. Similarly, run the prime factorization example on your own computer, and see what speedup you get.

Exercise 3. The lecture's construction of a histogram (counting the numbers of times that each prime factor 2, 3, 5, 7, 11 ... appears) uses a side effect in the assignment

```
histogram.[i] <- histogram.[i] + 1
```

But side effects should be avoided. Program the histogram construction so that it does not use side effects but purely functional programming. There are several useful functions in the Seq module. The final result does not have to be an int[] array, but could be a seq<int * int> of pairs (p, n) where p is a prime factor and n is the number of times p appears in the array of lists of prime factors.

Exercise 4. Find the fastest way on your hardware to count the number of prime numbers between 1 and 10 million; the answer should be 664579. Use this F# function to determine whether a given number n is prime:

```
let isPrime n =
    let rec testDiv a = a*a > n || n % a <> 0 && testDiv (a+1)
    n>=2 && testDiv 2;;
```

or if you believe it would be faster in C, C# or Java, you may use this version:

```
private static boolean isPrime(int n) {
  int k = 2;
  while (k * k <= n && n % k != 0)
    k++;
  return n >= 2 && k * k > n;
}
```

Remember to use parallel programming to the extent possible. On my Intel i7 Mac the result can be computed in 2 seconds wall-clock time using 4 cores, and 99 characters of F# code (in addition to the above function).