Portfolio element – Clojure

Unit	Programming languages: principles and design (6G6Z1110) Programming languages – SE frameworks (6G6Z1115)
Lecturer	Rob Frampton
Week	9
Portfolio element	Clojure (15% of coursework)

Introduction

This assignment is concerned with prime numbers. The following background information may be useful.

A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself. The property of being prime is called primality. A simple method of verifying the primarily of a given number *n* is as follows:

i) If $n = 1$, it is not prime
ii) If $n = 2$, it is prime
iii) Otherwise, is *n* a multiple of any integer between 2 and $\sqrt{n}$ inclusive? If so, then *n* is prime.

For example:

1) n = 1 is not prime, by definition
2) n = 101 is prime because 101 is not a multiple of any integer between 2 and $\sqrt{101} \sim 10.05$. That is: 101 is not a multiple of 2, 3, 4, 5, 6, 7, 8, 9 or 10.
3) n = 81 is not prime because it is a multiple of at least one integer between 2 and $\sqrt{81} = 9$. That is: 81 is a multiple of 3 and 9.

Assignment

In this assignment you will implement a set of Clojure functions which will be used to list the prime numbers within a certain range. You should implement the following:

a) Write a function named `get-divisors` which takes a number *n* as input and returns the all the numbers between 2 and $\sqrt{n}$ inclusive. For example:

```
   (get-divisors 4)
=> (2)
   (get-divisors 101)
=> (2 3 4 5 6 7 8 9 10)
```

b) Write a function named `divides?` which takes as inputs a divisor *x* and a number *n*. The function should return `true` if *x* divides *n* and `false` otherwise. For example:

```
   (divides? 2 10)
=> true
   (divides? 4 10)
```

```
=> false
```

c) Write a function named `no-divisors?` which takes an input *n*. The function should return `true` if **none** of the numbers between 2 and √*n* divide *n*, and `false` otherwise. The function should use both your `get-divisors` function and your `divides?` function.

Hint: you will probably need to wrap the `divides?` function in an anonymous function so that you can pass in the value of n.

For example:

```
   (no-divisors? 9)
=> false
   (no-divisors? 7)
=> true
```

d) Write a function named `is-prime?` which takes an input *n* and returns `true` if *n* is prime, and `false` otherwise. This function should check to see if *n* is 1 or 2 and respond accordingly; if not, it should call your `no-divisors?` function. For example:

```
   (is-prime? 1)
=> false
   (is-prime? 2)
=> true
   (is-prime? 3)
=> true
   (is-prime? 4)
=> false
   (is-prime? 101)
=> true
```

e) Write a function named `prime-seq` which takes inputs *from* and *to*. The function should return all the prime numbers between *from* and *to* inclusive. For example:

```
   (prime-seq 50 100)
=> (53 59 61 67 71 73 79 83 89 97)
   (prime-seq 7 11)
=> (7 11)
```

f) Write a function named `print-top-primes` which takes inputs *from* and *to*. This function should display the 10 largest primes in the range *from* and *to* inclusive, which should be obtained from the `prime-seq` function. It should then print out the sum of the 10 largest primes. For example:

```
   (print-top-primes 50 100)
97
89
83
79
73
71
67
61
```

```
59
53
Total=732
=> nil
   (print-top-primes 7 11)
11
7
Total=18
=> nil
```

Submission

You must submit a file named “Primes.clj” through Moodle. Submission link is available under the Week 9 section.