1. Make an inventory of the programming languages that you know, and how expert you are in each. Do you have a preferred language? Are there languages that you don’t know but would particularly like to learn?

2. In slide 12, the code relies on the statement `for d <- 2 until n` iterating over values up to, but not including `n`. If we wanted the range to include `n` as well, which keyword should we use instead of `until`? Try to answer this by looking up the Scala documentation pointed to from the course web page. (Only google if you must...)

3. The version of `isPrime` on slide 13 is purely functional, that is, it doesn’t depend on updateable variables or assignments. The behavior of pure functions can be understood as a simple calculation, just like in paper-and-pencil math. This gives us insight into how the program actually works. For example, we can calculate that

   ```scala
   def isPrime(n: Int): Boolean = 
     if (n <= 1) false 
     else !((2 to (n-1)).exists(d => n % d == 0)) 
   
   // Example: isPrime(5)
   isPrime(5) 
     = noDivFrom(2) [with n = 5 from now on] 
     = (2 >= 5) || (5 % 2 !=0) && noDivFrom(3) 
     = false || (5 % 2 !=0) && noDivFrom(3) 
     = (5 % 2 != 0) && noDivFrom(3) 
     = true && noDivFrom(3) 
     = noDivFrom(3) 
     = (3 >= 5) || (5 % 3 !=0) && noDivFrom(4) 
     = false || (5 % 3 !=0) && noDivFrom(4) 
     = (5 % 3 !=0) && noDivFrom(4) 
     = true && noDivFrom(4) 
     = noDivFrom(4) 
     = (4 >= 5) || (5 % 4 !=0) && noDivFrom(5) 
     = false || (5 % 4 !=0) && noDivFrom(5) 
     = (5 % 4 !=0) && noDivFrom(5) 
     = true && noDivFrom(5) 
     = noDivFrom(5) 
     = (5 >= 5) || (5 % 5 != 0) && noDivFrom(6) 
     = true || (5 % 5 != 0) && noDivFrom(6) 
     = true [because of short-circuit evaluation!]
   ```

Write down a similar calculation for `isPrime(9)`. 
4. Trace the behavior of the slide 24 stack machine on the following instruction sequence, assuming the initial values $a = 1$ and $b = 2$.

```
LOAD a
LOAD b
LOAD a
LOAD b
SUB
ADD
SUB
CONST 42
ADD
STORE a
```

Write down a simple statement in the style of the one on slide 24 from which this instruction sequence might have been generated.

5. In your own words, write down the difference between a compiler and an interpreter.