CS558 Programming Languages – Fall 2023 – Suggested Study Question Solutions for Lecture 1a

1. [No suggested solution.]

2. Using keyword `to` instead of `until` will have the desired effect. You can find this in Section 7.3 ("For expressions") of the "Programming in Scala" book.

3. Here's the analogous calculation:

```
isPrime(9)
= noDivFrom(2) [with n = 9 from now on]
= (2 >= 9) || (9 % 2 !=0) && noDivFrom(3)
= false || (9 % 2 !=0) && noDivFrom(3)
= (9 % 2 != 0) && noDivFrom(3)
= true && noDivFrom(3)
= noDivFrom(3)
= (3 >= 9) || (9 % 3 !=0) && noDivFrom(4)
= false || (9 % 3 !=0) && noDivFrom(4)
= (9 % 3 !=0) && noDivFrom(4)
= false && noDivFrom(4)
= false [because of short-circuit evaluation!]
```

4. Here's a trace:

```
                                    a = 1, b = 2
  LOAD a                  1         a = 1, b = 2
  LOAD b                2 1         a = 1, b = 2
  LOAD a              1 2 1         a = 1, b = 2
  LOAD b            2 1 2 1         a = 1, b = 2
  SUB                -1 2 1         a = 1, b = 2
  ADD                   1 1         a = 1, b = 2
  SUB                     0         a = 1, b = 2
  CONST 42             42 0         a = 1, b = 2
  ADD                    42         a = 1, b = 2
  STORE a                           a = 42, b = 2
```

This might have been generated from the statement `a = (a - (b + (a - b))) + 42`, if we followed the same left-to-right generation strategy as on the slide. But there are many other statements which might lead to the same stack code, and other stack code sequences that could be used to evaluate this statement.

5. There are many reasonable answers, but the essential point is that a compiler *translates* a program from a source language to a target language (usually because we already have a way to run programs in the target language), whereas an interpreter directly *executes* the source language program. Scott pg. 17 has some useful paragraphs.