Stream Coding Questions

Write out double_natural, which is a stream that evaluates to the sequence 1, 1, 2, 2, 3, 3, etc.
We assume flag is 1 initially

Restating the problem: We are given first. We want to create a stream that will represent the following sequence: first, first, first+1, first+1, first+2, first+2, …

Initial guess (ignore flag variable):
We want every element to be repeated twice. Then we need the following line in the body of function:

```
(cons-stream first (cons-stream first (double-naturals (+ 1 first))
```

We cons first two times – creating the sequence first, first – and then recursive call double-naturals incrementing first by 1

This solutions works! But is there a better way to do this? Can we do this without hardcoding in first two times? What if the problem was different: quadruple_natural? The amount of code we have to write increases immensely. How do we avoid this?

The first time we call the rest of double_naturals we want it to return first. The next time we call double_naturals we also want first to be returned. But the time after that we want first + 1. Think of it this way. Every two calls to the rest do the same thing. Then one condition changes (increment first) and we again return the same thing two times. Now we have this very basic idea of a solution:

```
If something:
      (cons-stream first …
      (cons-stream first …
```

We want the rest to computed through a call to double_naturals each time. So we know that we need a call to double_naturals as the rest of both the statements above:

```
If something:
      (cons-stream first (double_naturals FIRST_TOP FLAG_TOP))
      (cons-stream first (double_naturals FIRST_BOTTOM FLAG_BOTTOM))
```

We're almost there! Now, we know we return first two times before incrementing it. If the first time we need an element from double_naturals we go into the first line, and the next time we need an element we go into the second line, we have completed half of the task: we return first two times. We can do this with the help of flag. Flag corresponds to a condition that we check in the “if something” line. If flag is one thing go to line 1. If it is something else, go to line 2. We assumed flag was 1 initially so let's just have flag 1 correspond to going to line 1. This means that in our recursive call to double naturals in the first line we need to change flag to something else in order to make it go to line 2 the next time. Let's change it to 0 (this means FLAG_TOP = 0). What should FLAG_BOTTOM be if we want to go to line 1 after recursively calling double_naturals from the bottom line?

```
(if (= flag 1)
      (cons-stream first (double_naturals FIRST_TOP 0))         ← top line (1)
      (cons-stream first (double_naturals FIRST_BOTTOM 1))      ← bottom line (2)
```

Now we just need to find a place where we can increment first. We can't do it in the top line, this would mean that first becomes first+1 after the first access, which isn't what we want. Does it work if we increment first in the bottom line? Yes! So we get:

```
(if (= flag 1)
      (cons-stream first (double_naturals first 0))
      (cons-stream first (double_naturals (+ 1 first)  1))
)
```

*note: in the solutions, we do the opposite. We assume that flag is NOT 1 initially.