Recitation Session, November 15th, 2017

Please do not write on this sheet of paper
And do not use laptops during the session

Exercise 1

Consider the following series:

```
1
1 1
2 1
1 2 1 1
1 1 1 2 2 1
3 1 2 2 1 1
...........
```

1) Find the next element in the sequence above.

Now, let us encode an element of the sequence above as a `List[Int]`.

2) Write a function to compute the next element.

```
def nextLine(currentLine: List[Int]): List[Int] = ???
```

3) Implement a stream `funSeq` which constructs this sequence. Recall: to construct a stream, you can use `Stream.cons[A](a: A, b: => Stream[A]): Stream[A]`

```
lazy val funSeq: Stream[List[Int]] = ...
```

Exercise 2

1) Write a stream of squares of integers ≥ 1. You may use `Stream.from(i: Int)`

```
val squares: Stream[Int] = ...
```

2) Write a stream of all non-empty strings using the characters "0" and "1" and the concatenation operation +. In other words, every non-empty string composed of "0" and "1" should be reached at some point.

```
val codes: Stream[String] = ...
```

3) Using `codes`, write a stream of all possible non-empty palindromes of “0” and “1”. You may use the `.reverse` function defined on strings.

```
val palCodes: Stream[String] = ...
```

4) Can you do the same without filtering? The palindromes need not to be in the same order.

5) Given another stream `otherCodes`, possibly finite or infinite, you don’t know at first:

`val otherCodes: Stream[String] =` *[some external source]*

can you build a stream `allCodes` interleaving `palCodes` and `otherCodes` ?