CSYE 7200
Big-Data Sys Engr Using Scala
Assignment 2

I. *from* Method:

```
def from(start: Int, step: Int): ListLike[Int] =
                              LazyList(start, () => from(start+step))
```

II. Questions:

1.(a) what is the chief way by which *LazyList* differs from *Stream* (the built-in Scala class that does the same thing). Don't mention the methods that *LazyList* does or doesn't implement--I want to know what is the *structural* difference.
LazyList recursively generates the excessive elements.

(b) Why do you think there is this difference?
LazyList uses the byname parameters.

2. Explain what the following code actually does and why is it needed?
```
def tail = lazyTail()
```
Tail is to get the latest generated element in the list.
Because there is no real tail for an infinity list.

3. List all of the recursive calls that you can find in *LazyList* (give line numbers).
24, 41, 67,80,96,114,129,347,369,385

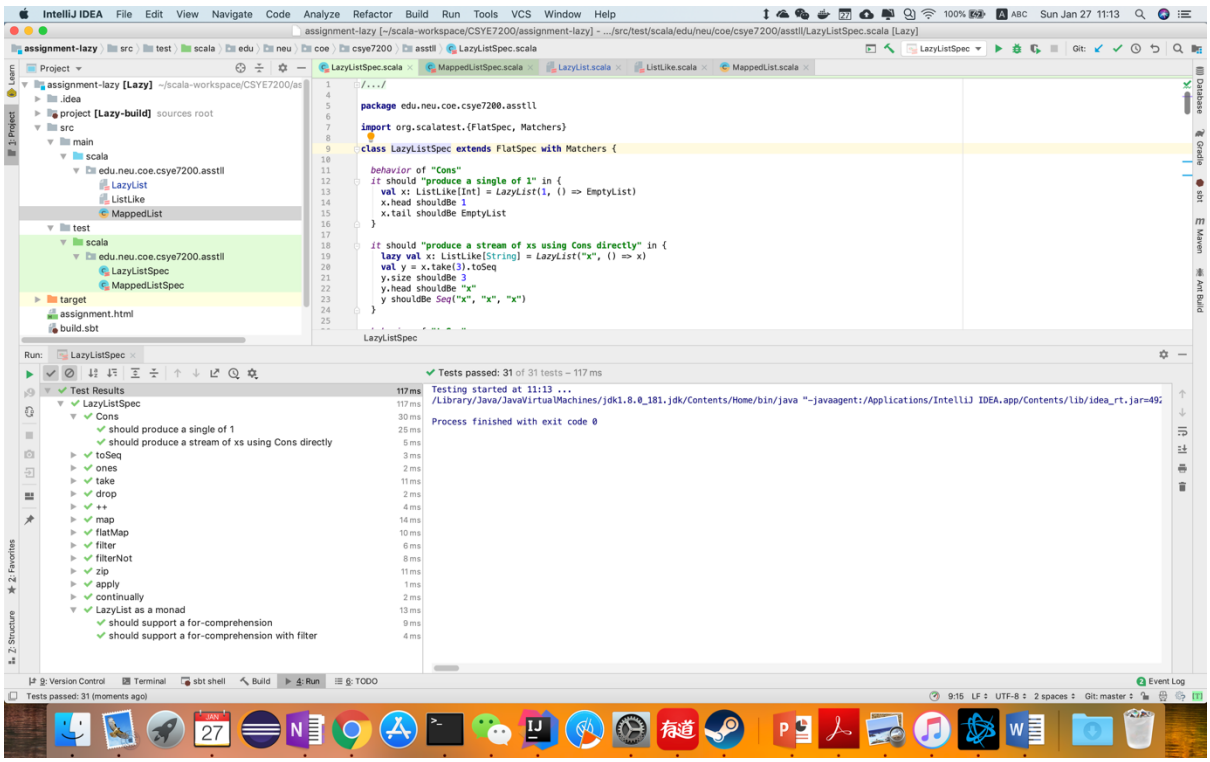
4. List all of the mutable variables and mutable collections that you can find in *LazyList* (give line numbers).
No mutable variables or mutable collections can be found.

5. What is the purpose of the *zip* method?
To zip 2 list ListX[x1, x2....] and ListY[y1,y2...] into a new list ListZip[(x1,y1), (x2,y2)...]

6. Why is there no *length* (or *size*) method for LazyList?
Because there is no real tail for an infinity list, so the length can be infinity.

III. Screenshots:

1. LazyListSpec



2. MappedListSpec