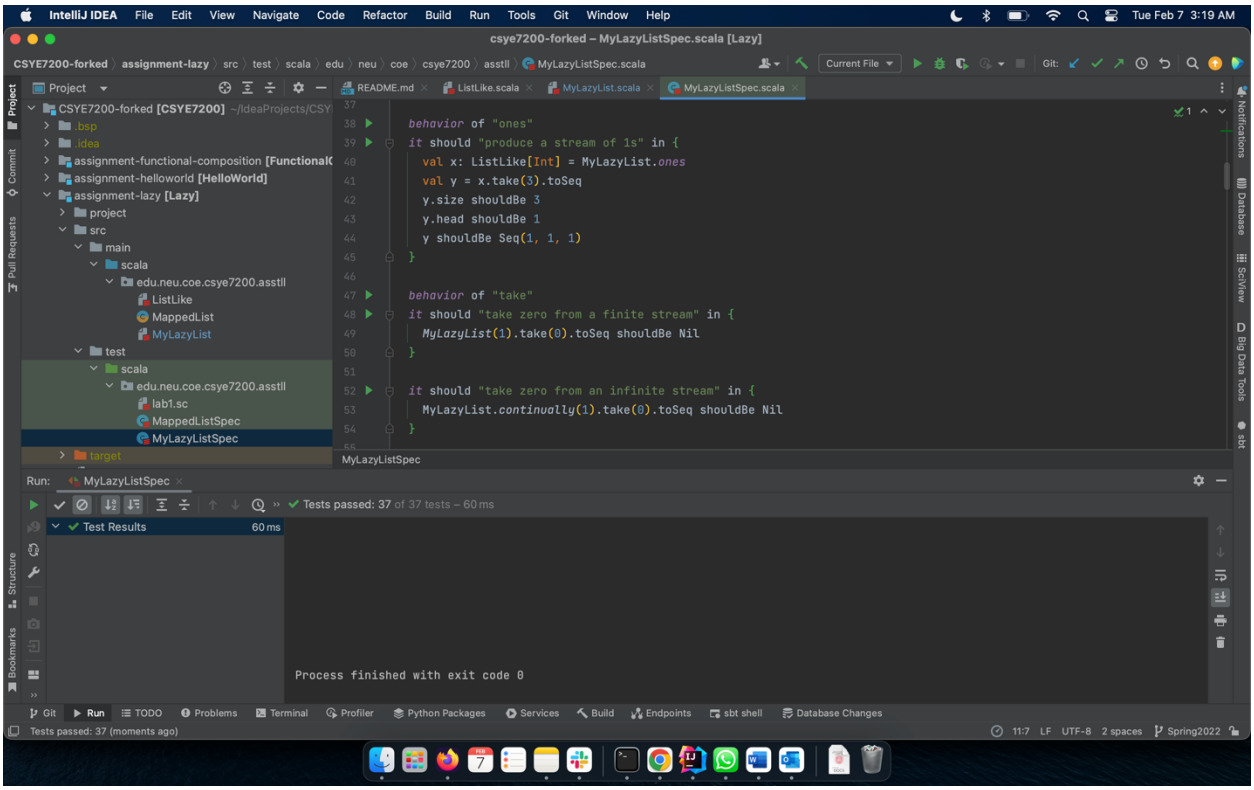
Sayeed Ahmed – SEC01 (NUID 002191535)

# Big Data System Engineering with Scala
# Spring 2023
# Assignment No. 2

Implementation of from method:
github link: https://github.com/sayeedahmed01/CSYE7200/tree/Spring2022/assignment-lazy/src/main/scala/edu/neu/coe/csye7200/asstll

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

Tests:




1. (a) what is the chief way by which *MyLazyList* differs from *LazyList* (the built-in Scala class that does the same thing). Don't mention the methods that *MyLazyList* does or doesn't implement--I want to know what is the *structural* difference.
   - The structural difference in the code is the way we have a abstract class that extends a trait to define a list structure and a couple of classes that are implemented for empty and non-empty lists

   (b) Why do you think there is this difference?
2. Explain what the following code actually does and why is it needed?

   ```
   def tail = lazyTail()
   ```
   - The code is used to get evaluate the tail of a List lazily i.e only when it is needed, this helps to optimize it for long lists where we evaluate the tail only when needed.

3. List all of the recursive calls that you can find in *MyLazyList* (give line numbers).
   - There are 7 recursive calls in *MyLazyList* :

1. Line 43: MyLazyList(y.head, () => y.tail ++ lazyTail().flatMap(f))
2. Line 82: case MyLazyList(y, g) => MyLazyList((x, y), () => lazyTail() zip g())
3. Line 361: case h :: t => MyLazyList(h, () => *apply*(t))
4. Line 372: def apply[X](x: X, xs: Seq[X]): ListLike[X] = MyLazyList(x, () => *apply*(xs))
5. Line 383: def continually[X](x: => X): ListLike[X] = MyLazyList(x, () => *continually*(x))
6. Line 408: def from(start: Int, step: Int): ListLike[Int] = new MyLazyList[Int](start, () => *from*(start + step, step))
7. Line 417: def
8.

4. List all of the mutable variables and mutable collections that you can find in *MyLazyList* (give line numbers).
    - There are no mutable variables or mutable collections

5. What is the purpose of the *zip* method?
    - The zip method is used to take two ListLike instances element by element and create a new instance of ListLike with the corresponding elements in a tuple.

6. Why is there no *length* (or *size*) method for *MyLazyList*?
    - There is no length method as the LazyList is supposed to be an infinite linked list, which would make the length method useless