Jiaqi Wang – SEC01## (NUID 001023711)

Big Data System Engineering with Scala
Fall 2021
Assignment No. 2

Screen shots

```scala
/**
  * Construct a stream of Integers starting with <code>start</code> and with successive elements being
  * greater than their predecessors by <code>step</code>.
  *
  * @param start the value of the first element.
  * @param step  the difference between successive elements.
  * @return a <code>ListLike[X]</code> with an infinite number of element (whose values are <code>x</code>,
  *         <code>x+step</code>, etc.).
  */
def from(start: Int, step: Int): ListLike[Int] =  MyLazyList(start, () => from(start+step, step)) // TO BE IMPLEMENTED
```

```
[info] - should produce a stream of even numbers using from(1)
[info] - should produce a stream of even numbers using from(2,2)
[info] filterNot
[info] - should produce a stream of even numbers using from(1)
[info] - should produce a stream of even numbers using from(2,2)
[info] zip
[info] - should zip together two empty streams
[info] - should zip together a stream and an empty stream
[info] - should zip together an empty stream and a stream
[info] - should zip together two non-empty streams
[info] apply
[info] - should produce a stream of a single 1
[info] continually
[info] - should produce a stream of 1s
[info] - should produce a stream of 1 thru 3
[info] LazyList as a monad
[info] - should support a for-comprehension
[info] - should support a for-comprehension with filter
[info] Run completed in 863 milliseconds.
[info] Total number of tests run: 62
[info] Suites: completed 2, aborted 0
[info] Tests: succeeded 62, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[success] Total time: 3 s, completed Sep 27, 2021, 2:37:01 AM
```

Github link: https://github.com/JiaqiWang1996/CSYE7200

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.
   (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()
```

3. List all of the recursive calls that you can find in *MyLazyList* (give line numbers).
4. List all of the mutable variables and mutable collections that you can find in *MyLazyList* (give line numbers).
5. What is the purpose of the *zip* method?

6. Why is there no *length* (or *size*) method for *MyLazyList*?

1.

(a)

In MyLazyList , tail is defined as the tail = lazyTail() which is a function, but in LazyList the tail is defined as tail: LazyList[A] = state.tail which is a lazylist;

MyLazyList has constructor and it's a case class ; the LazyList is not

MyLazyList class is like the cons class in Stream; the LazyList is a wrap class of cons

In MyLazyList class the method should finish its function but the LazyList class needn't

(b) by the usage of constructor , it can initialize an instance with methods; LazyList cannot do the same job

For MyLazyList , we have to evaluate each one till reaching the last one in order to reach the tail; As for LazyList, which is built with cons, can find tail using less memory. For MyLazyList, which is a case class, can be optimized when pattern maching.

Tail definition is totally different, former one is a function;latter one is a LazyList.

2.

It makes the class lazy evaluate, defines a method when invoked, then generates the tail of the lazylist.
We need it because we need to get the value of the lazyTail to use in some methods or test cases . It will return a list obj contains the rest part of the list which behind the head.

3.

98,116,131,361,383,408

4.

There is no mutable variables or collections cause the entire class is lazy.

5.

zip() method is a member of IterableLike trait, it is used to merge a collection to current collection and result is a collection of pair of tuple elements from both collections.

6.

In MyLazyList we are likely to have infinite number of elements in the list. If there is the length method, the stack can be overflow .