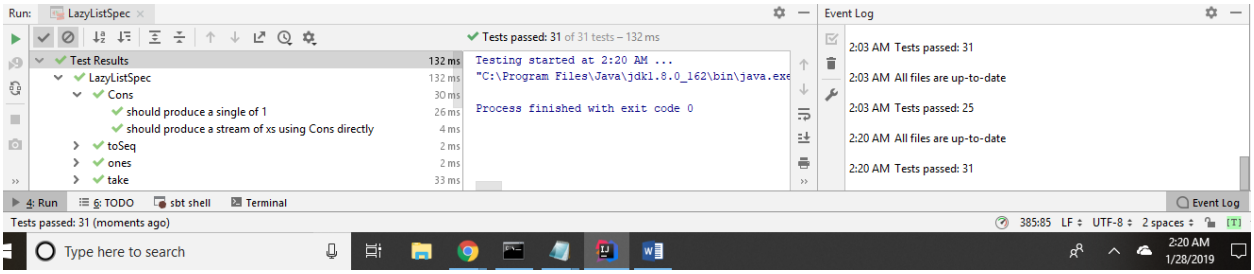
Assignment 2 (Lazy)

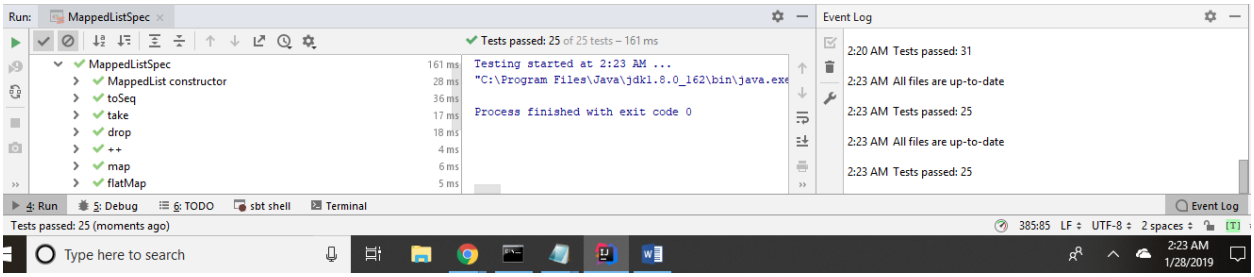
Name – Chitra Ramchand Paryani

NUID – 001869343

As per the assignment requirement, please find below the screenshots of the test cases passed in LazyListSpec.scala



Also, all the test cases passed in MappedListSpec.scala



Please find below screenshot of expression that I used to implement from method:

```
376     /**
377      * Construct a stream of Integers starting with <code>start</code> and with successive elements being
378      * greater than their predecessors by <code>step</code>.
379      *
380      * @param start the value of the first element.
381      * @param step  the difference between successive elements.
382      * @return a <code>ListLike[X]</code> with an infinite number of element (whose values are <code>x</code>,
383      *         <code>x+step</code>, etc.).
384      */
385     def from(start: Int, step: Int): ListLike[Int] = LazyList(start, () => from(start+step))
```

Additionally, please find below answer of questions asked in assignment:

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 the structural difference is.

   Scala stream is a subclass of a Scala sequence and does not need to be converted at all.

   A Scala sequence is simply a collection that stores a sequence of elements in specific order.

   A Scala list is a subclass of seq and a default implementation of sequence and to convert it into lazy we need to initialize list as lazy.

   When I am defining lazy list in REPL, I am getting output as below:
   List is not evaluating instead I am getting output as List[Int] = <lazy> (List of integer of type lazy)

```
scala> lazy val xs = List(1,2,3)
xs: List[Int] = <lazy>
```

2. While, when I am defining stream, I am getting output as below:

```
scala> val stream = Stream(1,2,3)
stream: scala.collection.immutable.Stream[Int] = Stream(1, ?)
```

   Stream is starting from 1 and ?

   When I am checking output of both list and stream, I am getting as below:

   List output:

```
scala> xs
res2: List[Int] = List(1, 2, 3)
```

   Stream output:

```
scala> stream
res3: scala.collection.immutable.Stream[Int] = Stream(1, ?)
```

```
scala> stream(1)
res7: Int = 2
```

```
scala> stream(2)
res8: Int = 3
```

The difference as I can be seen above, as I evaluated list – I got all the list output instantly where as in stream its evaluating output only when I am requesting, or we can say it only when it is required.

(b) Why do you think there is this difference?

I think this difference is because streams are lazy, non-strict and evaluate next value from stream only when it is required where as lazy list is evaluated when it is called as it is a default implementation of sequence that's why all elements are extracted sequentially one after the other.

3. Explain what the following code does and why is it needed?

```
def tail = lazyTail()
```

Here, we are defining tail having type ListLike[X] which when invoked returns tail of the list.

Why is it needed?

We are using it in flatMap function which flatten the results by concatenating all streams together.

FlatMap function returns a sequence for each element in the list and flattening the result in original list.

```
def flatMap[Y](f: X => Monadic[Y]): ListLike[Y] = {
  val y = f(x).asInstanceOf[ListLike[Y]]
  LazyList(y.head, () => y.tail ++ lazyTail().flatMap(f))
}
```

4. List all the recursive calls that you can find in LazyList (give line numbers).

I think below are the line numbers in which recursive calls are happening in LazyList - 24, 41, 58, 68, 80,96, 128 (tail recursion in inner method), 129, 162, 286, 347,358, 369, 374, 385

5. List all the mutable variables and mutable collections that you can find in LazyList (give line numbers).

All mutable variables and mutable collections are initialized using val in Scala.
Also, Scala by default always pick immutable collection.

Line Number – 40, 67, 374 defines mutable variables and mutable collections that I can find in LazyList

5. What is the purpose of the zip method?

Zip method is used on Mutable as well as Immutable collection data structure in Scala.

Zip method creates an Array of Tuple elements which is a merger of two original sequence.

So, if we have two collections and want to merge them as a tuple2 elements, so, we can use zip method.

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

   Here, in size method, we are using Scala's Option class, so, when a function succeeds, it returns an instance of Scala's Some class and when it fails, it returns an instance of Scala's None class.

   Here, in size method we are using return @Some(n) where n is the size of LazyList if it is definite and if size is not known lazy then return it to None.

   Also, in lazy, length method must be avoided every time as it can give bad runtime if list is infinitely long.

   So, that's why, here, we are checking if list is finite, then it returns value of the list else it returns None (Which means Null in Java)