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*?

Question1

a) In MyLazyList, we use case class to distinguish the empty or non-empty list.In LazyList we use state class to distinguish the empty or non-empty list.
In MyLazyList the method should finish it's function but in LazyList the method always just distinguish the empty or non-empty and then call an "impl" method to do the real function.
In MyLazyList the tail is defined as the tail = lazyTail() which is a function, but in LazyList the tail is defined as tail: LazyList[A] = state.tail whichi is a LazyList[A].

b) Because the class structural is totally different, in LazyList we don't have any case classes and each method should to distinguish the empty and call the "impl" method. But the MyLazyList don't have these structure.
In tail define, MyLazyList define as a function but in LazyList the tail is defined as a LazyList[A]. These are totally different.

Question2

Define a method when invoked, yields 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(like y.tail.head). If we don't define this method, it will cause some errors.

Question3

Line 131 case MyLazyList(h, f) => inner(rs :+ h, f())
Line 383 def continually[X](x: => X): ListLike[X] = MyLazyList(x, () => continually(x))
Line 408 def from(start: Int, step: Int): ListLike[Int] = MyLazyList(start, () => from(start + step, step))

Question4

There is no mutable variables or collections. Mutable variables are define by "var" and Mutable collections should import "scala.collection.mutable". Otherwise the variable and collections are immutable in Scala by default.

Question5

Zip method is used to merge a ListLike stream to current ListLike stream and result is a ListLike stream of pair of tuple element from both streams. When a streams is empty, this method will terminate.

Question6

In MyLazyList we could have infinte number of elements in the list. Lazy means that when we need it will be created. So we don’t know the length or the size of the MyLazyList, it’s indefinite.