Assignment 1 – Dry

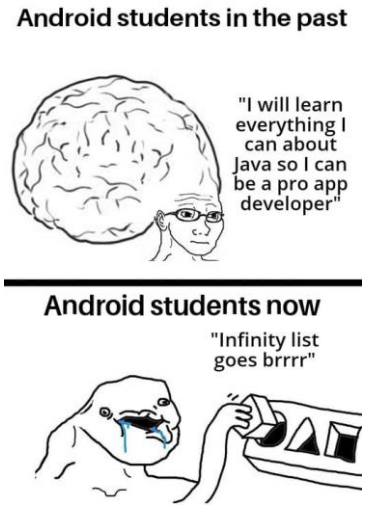
Exercise 1:

1. The 2 lines of code that make the list infinitely scrolling are:

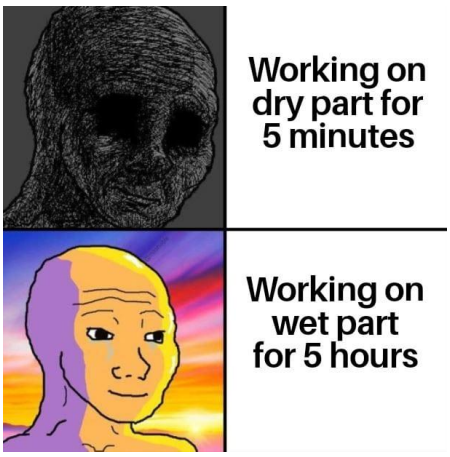
```
if (index >= _suggestions.length) {
  _suggestions.addAll(generateWordPairs().take(10));
}
```

If we remove these lines, and scroll to the end of the list, an error will be shown:

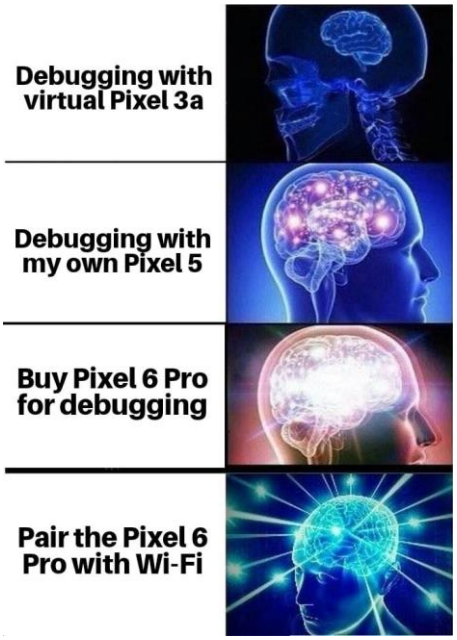
RangeError (index): Invalid value: Not in inclusive range 0..9: 10



2. A different method to construct such a list is to use "ListView.seperated()" constructor instead of the original "ListView.builder()" constructor. In my opinion, the different method is better because it has a special property "seperatorBuilder" for that purpose of building a new divider, which is more convenient and readable than checking if parameter i of itemBuilder is odd before creating a new divider.

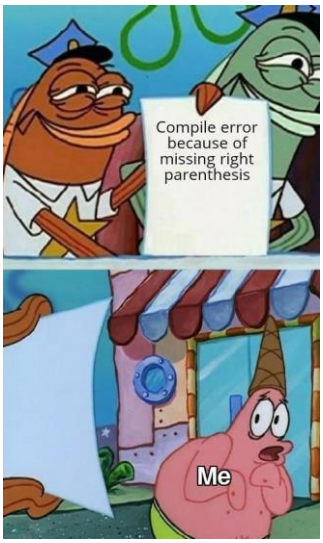


3. Calling setState notifies the framework that the internal state of this object has changed in a way that might impact the user interface in this subtree, which causes the framework to schedule a build for this State object. If we just change the state directly without calling setState, the framework might not schedule a build and the user interface for this subtree might not be updated to reflect the new state. Specifically in this app, the Saved Suggestions list uses setState to change the current items in the list, by removing or adding new pairs of words, according to the user's taps. Without setState, the user's taps won't change anything in the list.



Exercise 2:

1. The purpose of MaterialApp widget is to wrap some widgets that are commonly required for material design applications, and it configures the top-level Navigator to search for routes. Examples of its properties:
    - title - A one-line description used by the device to identify the app for the user.
    - color - The primary color to use for the application in the operating system interface.
    - theme - Default visual properties, like colors fonts and shapes, for this app's material widgets.



2. In general, the key property in all widgets controls how one widget replaces another widget in the tree. Specifically, The Dismissible widget has its own identity by the key property and used for example to update the tree whenever an item has been dismissed. It is a required property for Dismissible widget because it's a StatefulWidget.