Assignment 1

Id: 212234140

Date: 29/3/22

EXERCISE 1

1) The two lines of code that make the list infinitely scrolling are:
   If (index >= suggestion.length){
   _suggestions.addAll(generateWordPairs().take(10));
   }
   If we remove these lines, we will get range error if we would scroll down. This is because the suggestions list is only 10 lines long and we are trying to access the 11th element when scrolling.

2) Another method to construct such a list with dividers is with ListView.separated. on the one hand using the ListView.separated will be a cleaner coded then what we have now. But on the other hand, ListView.separated creates a fixed length and not infinite like what we have now.

3) Calling setState() inside the onTap() notifies the framework that the internal state of the ListTile 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 and show the changes.

EXERCISE 2

1) The purpose of the MaterialApp widget is to wrap a number of widgets that are commonly required for material design applications. Properties:
   a. **home** - It is used for the default route of the app means the widget defined in
   b. it is displayed when the application starts normally
   c. **theme –** It is used to provide the default theme to the application like the theme-color of the application.
   d. **Title –** It is used to provide a short description of the application to the user. For example, when the user presses the "recent apps" button on mobile the text proceeded in title is displayed.

2) A key is used when we are adding, removing, or reordering a collection of stateful widgets of the same type. Just like it done with the list of Dismissed widgets in our app. It is required because under the covers just like constructing a Widget tree, Flutter also builds an Element tree. The Element tree holds information about the type of each widget and reference to children elements. It can be thought as the skeleton of the app which controls the structures of it. When changing the order of the Widget tree (like removing an element or switching between two) Flutter walks the Element tree and checks that the corresponding widget in the Widget tree is the same *type. But when we use stateful Widgets the* information of the Widgets is being stored in state objects, not in the widgets themselves. So, comparing the type to determine if the Element tree should be changed too is not enough. It is needed to add a key to the widget which will identify it in a better way than just the type and will help flutter to know when a change in the elements tree is needed.