236272 HW1 - Dry

Exercise 1

1) The lines that make the list infinitely scrollable are:

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

Which check if the index of the item in the item-builder is bigger than the length of the word-pairs list. If it is, then we add another 10 random word pairs.
If we remove these lines, we get an out-of-bounds error when we scroll to the bottom since the _buildRow function tries to access the _suggestions list with an index bigger than the lists length.
If we initialize the _suggestions list with 10-word pairs and replace the body of the "if" statement with "return null" we get a scrollable list that stops when we get to the bottom.

2) Alternatively, instead of ListView.builder(), we can use ListView.seperated() to build our widget. In this case we need to add the itemCount property and set it to _suggestions.length, and also add the separatorBuilder property which is an anonymous function that gets the context and the index, and returns a divider:

```
separatorBuilder: (BuildContext context, int index) => const Divider(),
```

personally, I prefer this new method because it's built-in, its more readable, and it's easier to implement.

3) In a stateful widget, in order for us to see the changes (in our case its "saving" the word pair and changing the color of the icon) we need to call the setState function. The setState function notifies the framework that the state of the object has changed and so the framework schedules a new build.

Exercise 2

1) the method I used is the "push" method which pushes the next screen to the navigation stack:

```
Navigator.of(context).push(
```

another way is to use the Navigator.pushNamed method which gets the context and a named route and pushes it to the navigation stack:

```
// Within the `FirstScreen` widget
onPressed: () {
  // Navigate to the second screen using a named route.
  Navigator.pushNamed(context, '/second');
}
```

The named routes are defined in our MaterialApp widget in a property called "routes". We also need to add the initialRoute property to use this feature:

```
MaterialApp(
  // Start the app with the "/" named route. In this case, the app starts
  // on the FirstScreen widget.
  initialRoute: '/',
  routes: {
    // When navigating to the "/" route, build the FirstScreen widget.
    '/': (context) => FirstScreen(),
    // When navigating to the "/second" route, build the SecondScreen widget.
    '/second': (context) => SecondScreen(),
  },
);
```

when we navigate to a route, another widget is built and the pushNamed method puts it in the navigation stack.

2) Firstly, I created a snackbar widget with the requested message.
then, to show the snackbar I need to call showSnackbar which is a method of a ScaffoldMessenger widget (the other widget that is required in order to show the snackbar) which places the snackbar at the bottom of the screen:

```
final snackBar = SnackBar(
  content: Text('Deletion is not implemented yet'),
);  // SnackBar
ScaffoldMessenger.of(context).showSnackBar(snackBar);
```