Dry Exercise

Exercise 1

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

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

If we will remove the lines, and scroll to the end of the list we will get "out of range" error – that is because we will try to access a cell beyond the size of the _suggestions.
And as I assumed, when I tried to run the code with those changes I got this on the App:

```
RangeError (index): Invalid
value: Not in inclusive
range 0..9: 10
See also:
https://flutter.dev/docs
/testing/errors
```

2. A different method to construct such a list is to use the built in constructor of ListView: **ListView.separated**.
   This way is better for many reasons, such as:
   a. Much more readable code, you can easily understand where the separator is.
   b. Much easier to write, no need for any complicated logic as we used.

   All of the above is correct under the assumption of the known finite size of the list, and that because only under this assumption we can use this constructor.

3. As we learned, setState triggers the build function that renders the widget on the screen. We need to use setState to indicate that the current function will change the state of the app (from red heart to empty heart and vice versa), if we will not use it – those changed wont be shown to the user of the app.

Exercise 2

1. The method I used to implement navigation from the main screen to the logic screen was: **Navigator.push()** that is relative

   Another method that will give the same result can be:
   Navigator.pushNamed() that is absolute by the given name of the screen

2. The method I used to show the Snackbar was:
   ScaffoldMessenger.showSnackBar()
   the other widget that is required in order to show the Snackbar is Scaffold