HW 1 236272 DRY

Exercise 1:

1. The 2 lines that makes the list infinite is the lines:

```
if (index >= _suggestions.length) {
  // ...then generate 10 more and add them to
the
  // suggestions list.

_suggestions.addAll(generateWordPairs().take(10)
);
}
```

The change we need to do so that the app will show only 10 rows is initialize _suggestion like this:

```
final _suggestions =
generateWordPairs().take(10).toList();
```

And when we exceed from the list size return NULL:

```
if (index >= _suggestions.length) {
  // ...then generate 10 more and add them to the
  // suggestions list.
  return null;
}
```

After running the app we get the required result.

2. We used in ListView.builder but there is another option with using ListView.separated. Both of ways uses lazy building so that every time the user scrolls the page, the rows are generated. In our case where the list is finite and not containing a lot of rows there is no much use of that feature. But I think that the ListView.separated is better because it is more convenient to use, the dividers are created automatically and there is no need to deal with odd and even indices.

3. We need to call setState() inside onTap() because every time we tap on the heart icon we want that it will change to fully colored icon (red filling). so we need to call setState() that it job is to notify the framework that the internal state of this object has changed and regenerate the object(with the new icon).

Exercise 2:

1. I used the navigation.push method that push to the navigation stack the next screen (with MaterialPageRoute) and pop it when pressing back button. Another way is to use Navigator.pushNamed, in this way we define routes at the main class of the app like this:

```
    Return
MaterialApp(
                title: 'Flutter Demo',
                theme: ThemeData.light(),
                initialRoute: LoginScreen.route,
                routes: {
                  LobbyScreen.route: (context) => LobbyScreen(),
                  LoginScreen.route: (context) => LoginScreen(),
                },
              );
```

And when we want to change screen we call the method –

```
Navigator.pushNamed(context, LobbyScreen.route);
```

2. First, I created the snackbar -

```
final snackBar = SnackBar(
  content: Text('Login is not implemented
yet'),
);
```

And then I showed it by using this method –

```
// Find the ScaffoldMessenger in the widget tree
// and use it to show a SnackBar.
ScaffoldMessenger.of(context).showSnackBar(snackBa
r);
```

The other widget that required to show the snackbar is ScaffoldMessenger. This is widget that helps to show the snackbar at the bottom of the screen, to creates the visual structure and to ensures that important widgets don’t overlap.