Exercise 1

1.

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

If we remove these lines, then there wont be any treatment in situation of reaching the end of the available word pairings list and then on line after:
`return _buildRow(_suggestions[index]);`
the index would be out of range of existing pairings and there would be **range error** when scrolling to the end of the list on app.

2. ListView.separated

   I think this method is better because:

   a. The separator and the item are part of the same widget. it makes more sense to think about them as the same widget.
   b. The programmer has 2 designated constructures for these components, and he can use them straight forward instead of branching inside the same constructor in case index is odd or even. in addition he can style them in more sophisticated ways avoiding redundant branches.
   c. Code is more readable.

   Of course all the above is true in case list length is fixed because this method is supported only then.

3. We need the setState() because after pressing the heart icon the state of the word pairing get changed, from not-liked to liked (or vice versa) and we want action to take place in this situation – get in the liked suggestions list or get out of it.

Exercise 2

1. I used:

`Navigator.push()` – stack\relative.

We can use also: `Navigator.pushNamed` after defining route to login screen . stack but also absolute.

2. I used:

```
ScaffoldFeatureController<SnackBar, SnackBarClosedReason> showSnackBar(SnackBar snackBar)
```

for doing it we must use Scaffold widget.