Exercise 1 :

1. Identify the 2 lines of code that make the list infinitely scrolling:

```
if (index >= _suggestions.length) {
  _suggestions.addAll(generateWordPairs().take(10)); /*4*/
}
```

התנאי בודק האם מספר הפריט שאנחנו מוסיפים גדול מהמספר הקיים ברשימה, מהמספר הפריטים ברשימה, אם הוא יותר גדול אז הוא מייצר עוד פריטים חדשים לרשימה וככה זה מאפשר לרשימה להיות אינסופית.

What happens if we remove these lines, and scroll to the end of the list? Assume _suggestions start with a non-empty list of word pairs.

אם אנחנו מורידים את השורות האלה זה גורם למסך להיות עם שגיאה כי אין אפשרות להציג אינדקסים מסוימים והתוכנה לא יודעת מה לעשות שמגיעים לאינדקסים כאלה.

2. יש 4 אפשרויות לקונסרקטורים של listView :

- קונסטרקטור ברירת מחדל : מייצג באופן מפורש רשימה של ילדים list<widget> , דורש להגדיר את הילדים באופן מפורש, מתאים לרשימה עם מספר קטן של ילדים כי הוא בונה רשימה בשביל כל ילד אפשרי ברשימה במקום רק הילדים ברשימה.
- listView.builder : כמו בתוכנית בדוגמא , בונה את הילדים לפי דרישה ומתאים לרשימה עם כמות גדולה של ילדים או אינסופית כי הבנאי נקרא רק כאשר הילדים באמת נראים . עובד בצורה עצלה ולכן זאת הדרך הטובה ביותר כי אנחנו לא רוצים שזמן הפעלת התוכנית לראשונה תיקח הרבה זמן וגם בזמן ריצה אין פה הרבה בזבוז כי אנחנו קוראים לבנאי כל פעם כשילד שנוצר ולא לכמות גדולה של ילדים כל פעם.
- listView.separated – , בונה את הילדים לפי דרישה. המפריד בין הילדים נקרא בהתאם ומפריד בין הילדים. בנאי זה מתאים לרשימה עם מספר קבוע של ילדים.

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

- ListView.costom : הבנאי מתייחס לבן אחד silverChildDelegate שמספק את התכונות לילדים במודל.

כאשר יש לנו רשימה סופית ליצור עדיף לייצר אותה לפי הדיפולט כי מספר הפריטים ידוע מראש והמספר יחסית קטן אז עדיף שבזמן האתחול של האפליקציה זה יבנה ולא יתבזבז זמן ריצה.

3. צריך להשתמש ב setState() כדי להתאים את המצב החדש שאנחנו רוצים לפריט כאשר לוחצים על השורה עם הפריט המתאים, שינוי מצב . אם הוא לא היה מופיע לא היינו יכולים לשנות את המצב של משהו על המסך.

במקרה הזה אנחנו צרכים את ה setSet כי אנחנו רוצים שנלחץ על השורה עם הפריט זה ישנה את את הicon של הלב הריק ללב מלא.

Exercise 2 :

1. MaterialApp : A convenience widget that wraps a number of widgets that are commonly required for material design applications.
   Properties : (from the wet exercise)

- theme : Default visual properties, like colors fonts and shapes, for this app's material widgets.
- home : The widget for the default route of the app
- title : A one-line description used by the device to identify the app for the user.

2. The value of the key property use for know which item is swapping now.
   Each Dismissible widget should contain a Key to uniquely identify widgets.