1. Follow the lab lecture about **UI** tests then continue from step 2.

2. Download the **ListyCity** code from the following link. Git clone the repo below using the git commands you learned during Lab 4.
   https://github.com/Jakaria08/ListCity.git

3. In this Lab, we will use **Robotium** for UI testing. **Robotium** is an extension of the Android test framework and easy to use. Github repo link:
   https://github.com/RobotiumTech/

4. You can also use **Espresso**: https://developer.android.com/training/testing/espresso

5. To use Robotium, Include
   `androidTestImplementation 'com.jayway.android.robotium:robotium-solo:5.3.1`
   under *dependencies* on app Gradle (*build.gradle(Module:app)*) file and sync it.

6. We want to create UI tests for our **MainActivity** class as it is our UI class.

7. Under **com.example.simpleparadox.listycity(androidTest)** folder, create a new class **MainActivityTest** to test the UI functionalities.

8. We use the `'ActivityTestRule'` class to create a `rule`. From the official Android docs:

   "*This rule provides functional testing of a single `Activity`. When `launchActivity` is set to true in the constructor, the Activity under test will be launched before each test annotated with `Test` and before methods annotated with `Before`, and it will be terminated after the test is completed and methods annotated with `After` are finished.*"

9. We set the `initialTouchMode`: true -- The Activity would be placed into "touch mode" when started.

10. The main class for testing with Robotium is Solo. Solo is initialized with the instrumentation of the test case and the first activity to test.

```
package com.example.simpleparadox.listycity;

import androidx.test.rule.ActivityTestRule;
import com.robotium.solo.Solo;
/**
* Test class for MainActivity. All the UI tests are written here. Robotium test framework is
used
*/
public class MainActivityTest {
```

```
    private Solo solo;


@Rule
public ActivityTestRule<MainActivity> rule =
        new ActivityTestRule<>(MainActivity.class, true, true);
    }
}
```

11. Then, add the following methods:

```
/**
* Runs before all tests and creates solo instance.
* @throws Exception
*/
@Before
public void setUp() throws Exception{
    solo = new Solo(InstrumentationRegistry.getInstrumentation(),rule.getActivity());
}
/**
* Gets the Activity
* @throws Exception
*/
@Test
public void start() throws Exception{
    Activity activity = rule.getActivity();
}
```

`setUp()` method runs before every test using *@Before*. This method used to create the solo object with instrumentation and activity as arguments. The Instrumentation allows the test case to subscribe to various application events (key presses, etc), and also programmatically control the UI to enable functional testing of your application. Here the first test is `start()` method which gets the MainActivity.

12. Then add the first test for testing our MainActivity. We want to check the listview by adding a new city and also check the listview after clearing all data. Follow the comments for details.

```
/**
* Add a city to the listview and check the city name using assertTrue
* Clear all the cities from the listview and check again with assertFalse
*/
@Test
public void checkList(){
// Asserts that the current activity is the MainActivity. Otherwise, show “Wrong Activity”
```

```
  solo.assertCurrentActivity("Wrong Activity", MainActivity.class);

  solo.clickOnButton("ADD CITY"); //Click ADD CITY Button

  //Get view for EditText and enter a city name
  solo.enterText((EditText) solo.getView(R.id.editText_name), "Edmonton");
  solo.clickOnButton("CONFIRM"); //Select CONFIRM Button
  solo.clearEditText((EditText) solo.getView(R.id.editText_name)); //Clear the EditText

  /* True if there is a text: Edmonton on the screen, wait at least 2 seconds and find
minimum one match. */
  assertTrue(solo.waitForText("Edmonton", 1, 2000));

  solo.clickOnButton("CIEAR ALL"); //Select CIEAR ALL
  //True if there is no text: Edmonton on the   screen
  assertFalse(solo.searchText("Edmonton"));
}
```

13. Add a city to the list and get the first item from the list. Then check the name. Follow the comment for details.

```
/**
* Check item taken from the listview
*/
@Test
public void checkCiyListItem(){
  solo.assertCurrentActivity("Wrong Activity", MainActivity.class);

  solo.clickOnButton("ADD CITY");
  solo.enterText((EditText) solo.getView(R.id.editText_name), "Edmonton");
  solo.clickOnButton("CONFIRM");
  solo.waitForText("Edmonton", 1, 2000);

  // Get MainActivity to access its variables and methods.
  MainActivity activity = (MainActivity) solo.getCurrentActivity();
  final ListView cityList = activity.cityList; // Get the listview
  String city = (String) cityList.getItemAtPosition(0); // Get item from first position
  assertEquals("Edmonton", city);
}
```

14. Finally, add `tearDown()` method using the *@After* tag to run after every test method. This method closes the activity after each test.

```
/**
* Closes the activity after each test
* @throws Exception
*/
@After
public void tearDown() throws Exception{
    solo.finishOpenedActivities();
}
```

15. Completed MainActivityTest class:

```
package com.example.simpleparadox.listycity;

import android.app.Activity;

import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.platform.app.InstrumentationRegistry;
import androidx.test.rule.ActivityTestRule;
import android.widget.EditText;
import android.widget.ListView;

import com.robotium.solo.Solo;

import org.junit.After;
import org.junit.Before;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

import static junit.framework.TestCase.assertTrue;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertFalse;
/**
* Test class for MainActivity. All the UI tests are written here. Robotium test framework is
used
*/
@RunWith(AndroidJUnit4.class)
public class MainActivityTest{

    private Solo solo;

    @Rule
    public ActivityTestRule<MainActivity> rule =
```

```
            new ActivityTestRule<>(MainActivity.class, true, true);
    /**
     * Runs before all tests and creates solo instance.
     * @throws Exception
     */
    @Before
    public void setUp() throws Exception{

        solo = new Solo(InstrumentationRegistry.getInstrumentation(),rule.getActivity());
    }
    /**
     * Gets the Activity
     * @throws Exception
     */
    @Test
    public void start() throws Exception{
        Activity activity = rule.getActivity();
    }
    /**
     * Add a city to the listview and check the city name using assertTrue
     * Clear all the cities from the listview and check again with assertFalse
     */
    @Test
    public void checkList(){
      //Asserts that the current activity is the MainActivity. Otherwise, show “Wrong Activity”
        solo.assertCurrentActivity("Wrong Activity", MainActivity.class);

        solo.clickOnButton("ADD CITY"); //Click ADD CITY Button

        //Get view for EditText and enter a city name
        solo.enterText((EditText) solo.getView(R.id.editText_name), "Edmonton");
        solo.clickOnButton("CONFIRM"); //Select CONFIRM Button
        solo.clearEditText((EditText) solo.getView(R.id.editText_name)); //Clear the EditText

        /* True if there is a text: Edmonton on the screen, wait at least 2 seconds and
        find minimum one match. */
        assertTrue(solo.waitForText("Edmonton", 1, 2000));

        solo.clickOnButton("CIEAR ALL"); //Select CIEAR ALL
        //True if there is no text: Edmonton on the   screen

        assertFalse(solo.searchText("Edmonton"));
```

```
    }

    /**
     * Check item taken from the listview
     */
    @Test
    public void checkCiyListItem(){
        solo.assertCurrentActivity("Wrong Activity", MainActivity.class);

        solo.clickOnButton("ADD CITY");
        solo.enterText((EditText) solo.getView(R.id.editText_name), "Edmonton");
        solo.clickOnButton("CONFIRM");
        solo.waitForText("Edmonton", 1, 2000);
        // Get MainActivity to access its variables and methods.
        MainActivity activity = (MainActivity) solo.getCurrentActivity();
        final ListView cityList = activity.cityList; // Get the listview
        String city = (String) cityList.getItemAtPosition(0); // Get item from first position
        assertEquals("Edmonton", city);
    }

    /**
     * Close activity after each test
     * @throws Exception
     */
    @After
    public void tearDown() throws Exception{
        solo.finishOpenedActivities();
    }
}
```