Name: _____

ROSE-HULMAN INSTITUTE OF TECHNOLOGY **CSSE483 Android Application Development**

Lab 1: Getting Started with Buttons

The purpose of this lab is to give you some practice building simple Android apps.

_____ Part A: Building a Simple User Interface

_____ Part B: Linear Lights Out

A. Building a Simple User Interface

1. Create a new Application called MyFirstApp*YourName*. You can use your username or initials as long as they are unique among your classmates.) Setup the UI for the MainActivity using the link below. Note that the link is to a **single step** in a Google tutorial. Only do that one step. You do ***not*** need to do the "Starting Another Activity" step that comes next. So only this one page.
http://developer.android.com/training/basics/firstapp/building-ui.html
Complete that page then use the instructions below.

2. So far so good, but the button doesn't do anything. Add java code to listen for the button press. If it is pressed, it checks to see what the user entered. If they entered "secret", the button's text will change to "Wow". Otherwise, it stays, "Send" like in the tutorial.

Hint: The body of the onClick method will include this:

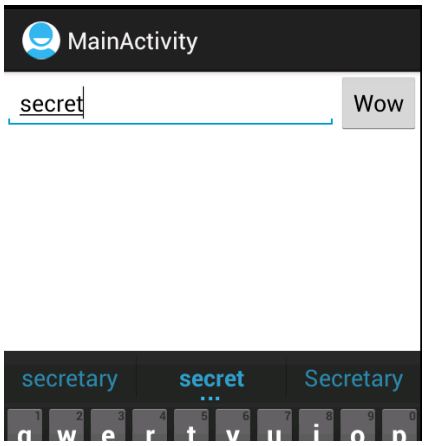
```
if (editText.getText().toString().equals(getString(R.string.secret))) {
        button.setText(R.string.button_text_wow);
} else {
        // you fill this in
}
```

Notes: (1) Check out the return type for the getText() method. Can you see why you need the **toString()**? (2) Note that the setText() method can take a resource

ID (a string from strings.xml) as a parameter. (3) I even made "secret" a string in strings.xml. Putting all your strings in strings.xml is good practice that you should follow.

Note: When you copy/paste the code, you'll need to press Ctrl-Shift-O to import the necessary packages. Sometimes you'll need to select the Android version of the package (usually the top option). For the edit box area make sure you import the android.view.**View.**OnKeyListener ***not*** the android.content.**DialogInterface**.OnKeyListener.

Your final app should look like this:



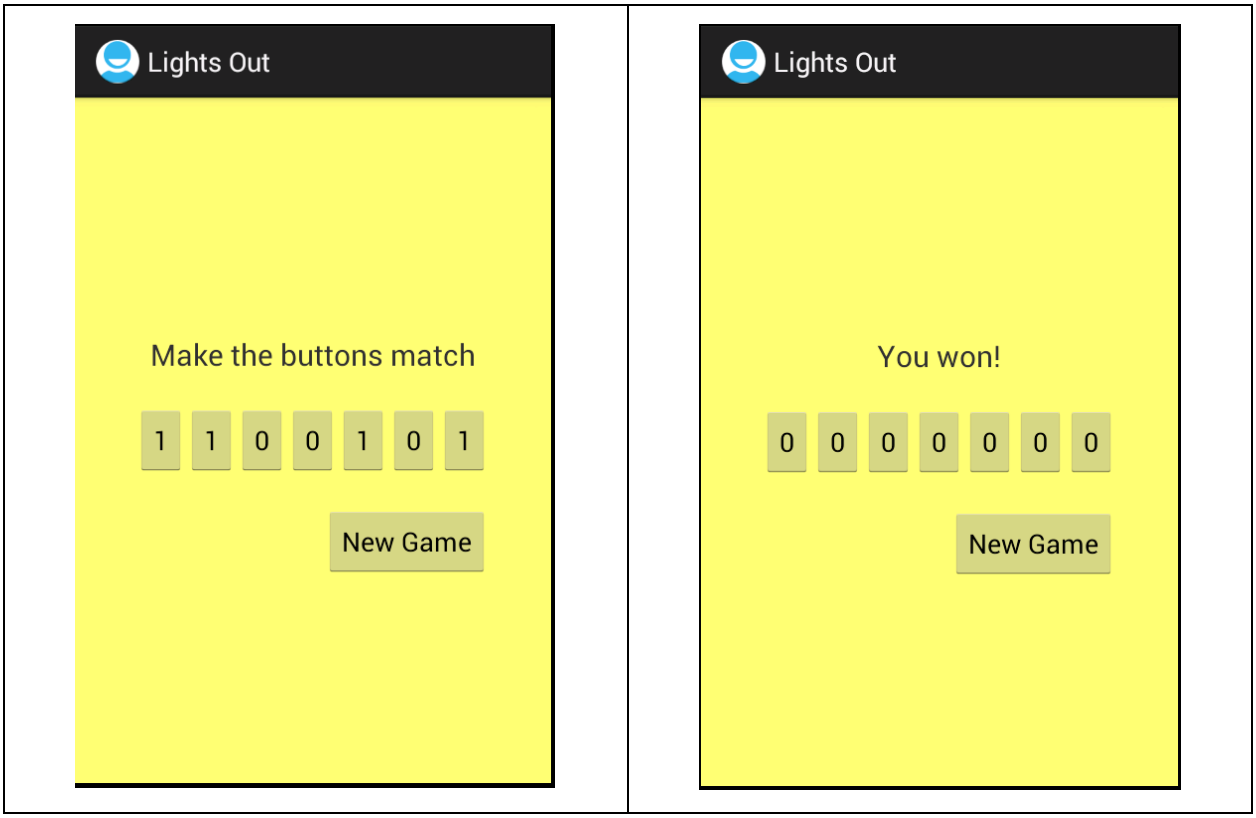
To get credit for this part of the lab you will need to show your emulator (or device) running this app to an instructor.

B. Linear Lights Out

App name: LightsOut*YourName.*

The purpose of this app is to learn about creating a view using an XML layout file then connecting it to a model object. The game you will be developing is called *Lights Out*. The goal of Lights Out is usually to click lights and eventually turn them all off, hence the name "Lights Out". When a user clicks a light it toggles the state of the light that was clicked and toggles the state of all neighbor lights. Instead of lights we'll just use 0 and a 1 and try to make all the buttons match.

In our version we will have a line of exactly **seven** buttons arranged in a line. We fix the number to keep it simple; in a future version, we will make the number of buttons variable.



Note that clicking on the first or last button in the line toggles only two buttons. Clicking any other button will toggle three buttons.

The button row and text should be centered. The text size should be 20 sp. The new game button should be right-aligned with the buttons. You can choose reasonable margins:

To help you out you can download the model object called LightsOutGame.java (or

you can make it yourself).
Model object backing class: LightsOutGame.java

There are really only three functions you need to use from that class:

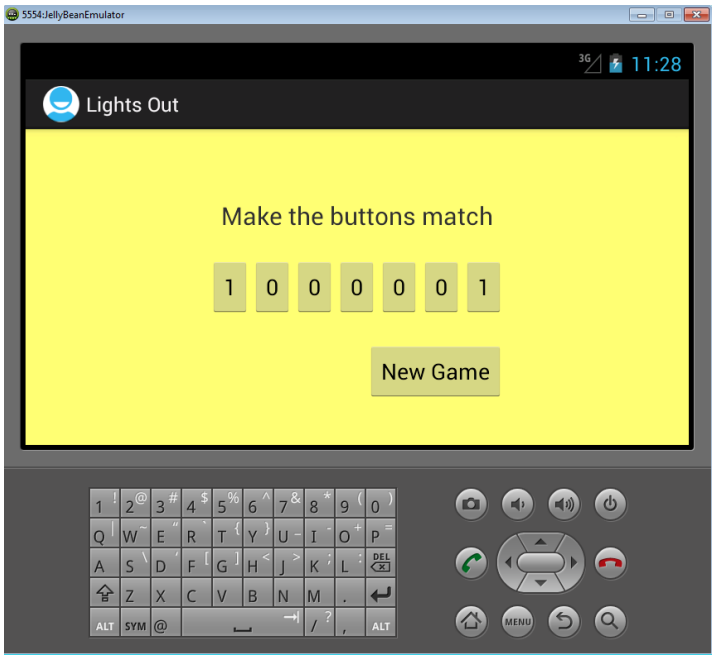
```
public LightsOutGame(int numButtons);
public boolean pressedButtonAtIndex(int buttonIndex);
public int getValueAtIndex(int buttonIndex);
```

Here is the javadoc if you prefer: LightsOutGame Javadoc

These should be pretty straightforward. The boolean value returned from pressedButtonAtIndex determines if the game was won with that click. If the game was won then you should change the text to say "You Win!" (or something like that). If the game was not won with that click display "Make the buttons match" (or something like that). Note, you may need to change the package name for that file when you add it to your project to make it match your package name.

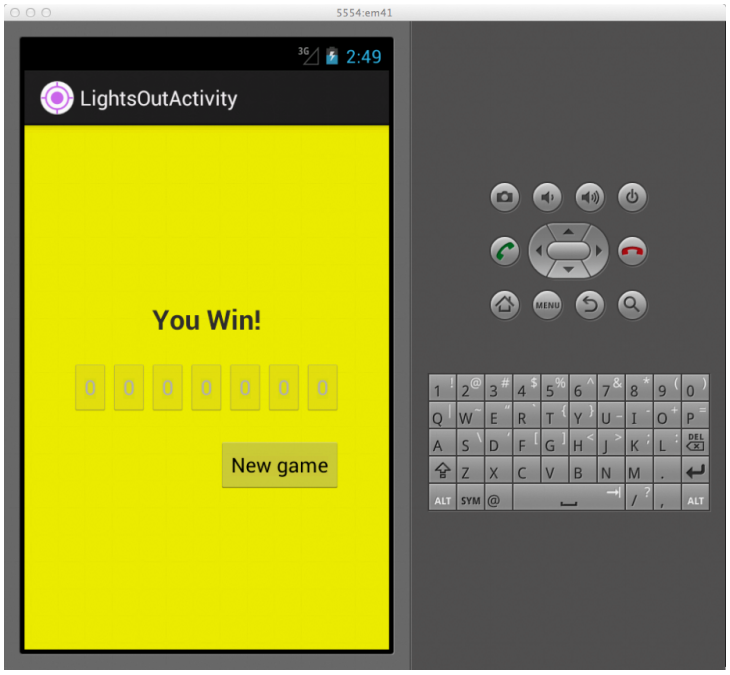
One note: at first my buttons were too wide to show all seven in portrait mode. If you view a Button in the Properties view, it has a minWidth - mine was 64dp. (Interestingly, it doesn't show up in the layout.xml, but in values > dimens.xml). Changing these to 32dp fixed it for me.

With only 7 buttons it should be fairly easy to make the layout look good on most emulator screen sizes in landscape or portrait (rotate emulator using ctrl-F12). Make the background color anything other than black.



Note: It is **okay** to **NOT** save state between launches and rotations. We will cover saving state in a future unit.

As a final step disable all the game buttons when a player wins. Then re-enable them when a new game starts. When the buttons are disabled they can't be clicked and they will gray out slightly. (Hint: I expect you to search the javadoc for buttons to find the method to do this.) Note that the "New Game" button should always be enabled.



When you finish this project show your instructor! You finished Lab 1.

Submission Note: If you are taking this course online, to show your instructor, you should capture a video of you launching the app on your phone or emulator, and then playing the game until you win, resetting the game using the new game button to show that it randomizes again, and then winning the game a second time. Hold the phone nice and close to the camera so I can see it. And please explain what you are doing as you go loudly and clearly - since I don't get to see you face-face, I especially want to hear your voice! :) Submit the video along with your zipped Eclipse project.