COMP3074 Fall 2016 Assignment 1

Objective: Practice basic functionality on Android

Evaluation: This assignment is worth 15% of your final grade

Marking: This assignment will be marked out of 15 points as follows:

- Functionality: the app works on a basic emulator on Android Studio - 5 marks
- Features: as described below - 5 marks
- Quality of the code - 5 marks

Due Date: Oct 8, 2016 by midnight, on Blackboard.

Late submissions: Will be penalized 10% per day until your mark reduces to 0. Even a few minutes after midnight it is considered one day.

Submission format: A zipped copy of the app folder.

Academic Honesty: Your Java code will be checked by software (https://theory.stanford.edu/~aiken/moss/) for authenticity.

Learning Goals:

In this assignment, we shall learn something about

- Defining colors in Android
- Combinations of *LinearLayouts*
- Implementing *Buttons*
- Defining array resources in XML
- Using interfaces to add functionality to a class
- Detecting and processing events
- Using anonymous inner classes to handle events

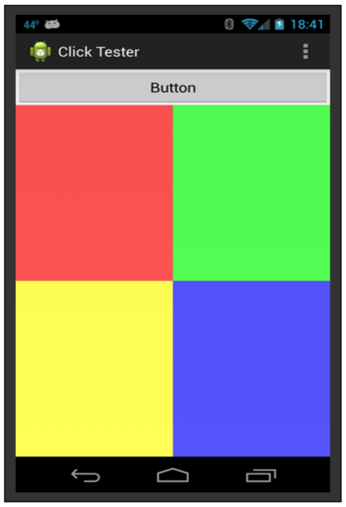
Create a new project named ClickTester as follows:

Application Name: ClickTester
Company Domain:< *YourNamespace* >
Package Name: <*YourNamespace*> . clicktester
Project Location: <ProjectPath> ClickTester
Target Devices: Phone and Tablet; Min SDK API 15
Add an Activity: Empty Activity
Activity Name: MainActivity (check the *Generate Layout File* box)
Layout Name: activity_main

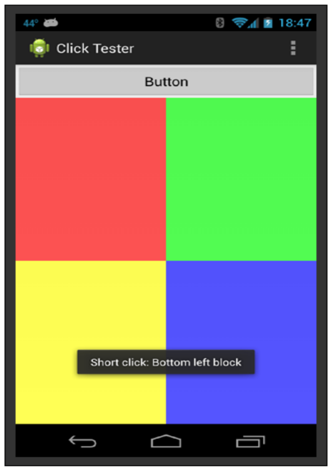
Create a layout as follows:

Much of this should be familiar from the layout for the previous *WebViewDemo* project (except that there we used a *RelativeLayout* but here a *LinearLayout* will be employed):

1. The parent container is a *LinearLayout*, with an *android:orientation="vertical"* property, so its children will be arrayed vertically.
2. The parent container has three children: one *Button* and two additional *LinearLayouts*. Each of the child*LinearLayouts* has an *android:orientation="horizontal"* property, so they will array their children horizontally.
3. Each of the child *LinearLayouts* contains two *Views*, and since each of the *Views* has the property *android:layout_weight="1"* set, each should have equal area weight in the layout.
4. The attribute value *fill_parent* suggests that an object expand to fill its parent container (minus padding), while the attribute value *wrap_content* means that the object wants to be just big enough to enclose its content plus any padding.
5. The properties like *android:background="@color/redback"* set color backgrounds for the four *Views*, with *@color/colorname* referencing the color *colorname* defined previously in *app/res/values/colors.xml*.



Implement event handling so if you execute the app a long or short press on any of the five widgets in our layout should produce an appropriate *Toast* display indicating the nature of the action. The figure below illustrates: A short finger tap on the yellow block at the lower left has just been completed. So we see that by this basic approach of laying out widgets and attaching event listeners we can listen for short and long finger taps on the display, sort out which widget was tapped, and take a corresponding action.



A click on the button must terminate the app.