CPEN 475/575
Project 1: Implement an alertness test program

Deliverables:

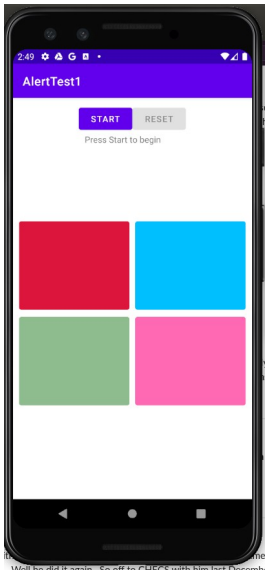
Please clean your Android Studio project and then zip the containing directory. Please include your name in the name of the zip. For example Proj1_Keith_Perkins.zip

Target SDKs

I will compile and test it on a Pixel 3 running API 29. Please target a minSdkVersion of 24 and a targetSdkVersion of 30 in your build.gradle file.

Description

You are to implement an android program which tests user alertness. Once started the user will be presented with a GUI similar to the following;



Once Start is clicked the app will randomly instruct the user to touch one of the 4 colors in the center. If the user touches the correct color the ‘Number right’ text view is incremented, otherwise the ‘Number Wrong’ view is incremented.

This process repeats 2 times and is not adjustable (this is a brittle design BTW, a better implementation would have a settings menu to adjust this number, we will get to that). Once complete the program evaluates the user based on the Number Right.

The following bit of code is how I did this;

```
private void evaluateUser() {
    if (numberCorrect == NUMBER_ROUNDS)
        myString = getString(R.string.Perfect_Score);
    else
        myString = getString(R.string.Pay_Attention);
    feedback.setText(myString);
}
```

Note the *NUMBER_ROUNDS* constant. You can tell it is intended to be a constant because it is all caps and convention states that variables that are all caps are constants. Fortunately we are the sort of people that hew to convention and strive to write clean code. So define this constant as follows in your activity;

```
private static final int NUMBER_ROUNDS = 2;
```

Also please define all of your strings in the strings.xml resource file. This is the best way to do it, and in the real world, the only way. The string Perfect_Score is pulled from a resource xml file (located in the values/strings.xml). The advantage is that all strings are located in one place, which greatly simplifies proofreading and language translation.

```
      myString = getString(R.string.Perfect_Score);
```

Here are the strings I used for this project;

```
<resources>
    <string name="app_name">AlertTest1</string>
    <string name="menu_settings">Settings</string>
    <string name="dont_walk">You probably should not walk</string>
    <string name="hello">Goodby</string>
    <string name="NumbRight">"Number Right = "</string>
    <string name="NumbWrong">Number Wrong =</string>
    <string name="Perfect_Score">Splendid, a perfect score</string>
    <string name="Pay_Attention">Please pay more attention</string>
    <string name="Press_Red">Press Red</string>
    <string name="Press_Blue">Press Blue</string>
    <string name="Press_Green">Press Green</string>
```

```
    <string name="Press_Pink">Press Pink</string>
    <string name="ERROR">ERROR</string>
    <string name="press_start_to_begin">Press Start to begin</string>
</resources>
```

I also added the following colors to the res/values/colors.xml file and used them to color the buttons in the UI.

```
<color name="Pink">#FF69B4</color>
<color name="Red">#DC143C</color>
<color name="Green">#8FBC8F</color>
<color name="Blue">#00BFFF</color>
```

Stuff to do:
Please give the project an icon of your choice.
Please carefully observe how the start and reset buttons affect the UI.
Please ensure you replicate this behavior.
For required behavior, please install the apk given on the project website and produce an app that mirrors its functionality.
Please ensure your application properly handles device rotations

Files you probably need to modify:
Mainactivity.java
activity_main.xml
colors.xml
strings.xml
some/all of the icon files

Grading:
%5 Follow submission directions
%5 icon correct (both in app drawer and on desktop)
%20 enable/disable behavior correct
%10 colored buttons implemented correctly
%10 strings and colors pulled from resource files
%20 App state implemented correctly
%30 Rotations handled correctly

Possible Gotchas:
If the user has entered 2 values, stop tracking whether they are right in the UI. Force them to reset to start again.

Additional Help:
The colored squares are buttons with the background color changed. (To change a button color, you need to add a single line of code in the XML i.e. app:backgroundTint="@color/red")

You can manipulate the size of the buttons using the `layout_weight` attribute in the xml layout file for any of the viewgroups.

I used both a linear layout and a table layout in my xml layout for this project. If all else fails color your layouts so you can see what area they cover. To change the background color of both viewgroups (like LinearLayout) and views (like button), use the following syntax in the view or viewgroups definition;

```
android:background="@color/Aquamarine"
```