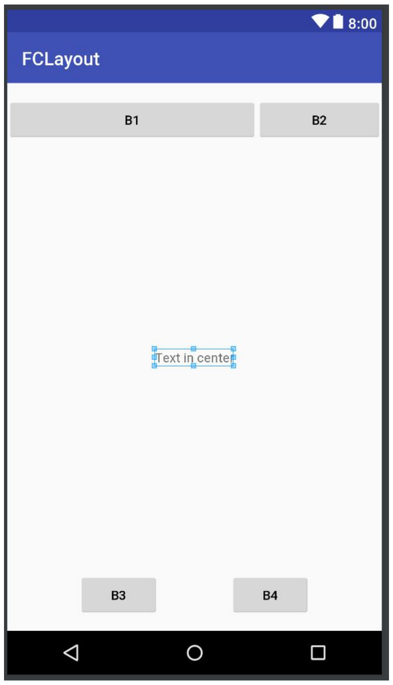
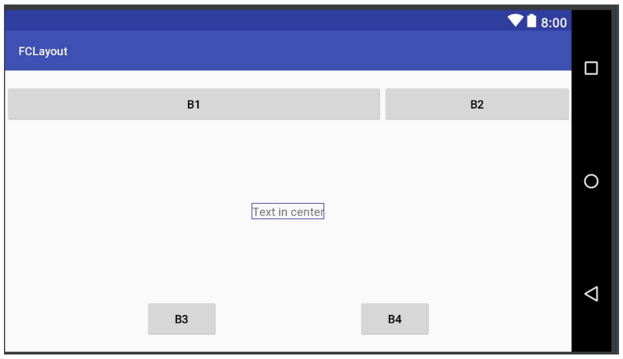
CS 371M: Flipped classroom 0: Layout

Recall the rules for flipped classrooms: do the work alone or with a partner that is unique for the semester. You can find the github link on piazza and via github classroom.
`https://classroom.github.com/classrooms`

Only one student needs to submit this code, but both students may do so if they wish. This assignment should be submitted through github (sorry, partners can't share the repository). Include a README at the top directory level that contains both student's EIDs.

The purpose of this exercise is to lay out four buttons (named B1, B2, B3, and B4) and one text view. You will do two independent layouts that all look (almost) the same. One uses a Linear Layout (and possibly nested Linear Layouts), the other uses a Constraint Layout (no nesting needed). The layouts for Constraint and Linear should be pretty much identical. Here is a visual depiction of your goal. Also see the (short) video, accessible via the course web page.





Above are portrait and landscape screenshots of our ideal layout, with the two top buttons 16dp from the toolbar, aligned with each other, with the left button (B1) $\frac{2}{3}$ of the horizontal real estate relative to the parent (root window), and the right (B2) the remaining $\frac{1}{3}$. The buttons can touch. The text is in the middle of the entire screen. The other two buttons (B3 and B4) are 16dp from the bottom and equidistant from each other and from the edges of the phone. They are wrap_content in horizontal size.

Your goal is to get as close as you can to this layout, **without** using fixed-sized directives. The one exception is the 16dp margin, which you are allowed to specify for TWO view objects (buttons or layouts). Another exception is the size 0dp, because that often does not act like a fixed width. Please remember, if you are specifying widths using explicit numbers like

160dp, then you are doing it wrong. Note that explicit numbers for weights and biases are fine because those values express a ratio.

We suggest writing the layout while using the live preview feature of Android Studio.

We do not provide either layout file, you have to write them from scratch. Feel free to have Android Studio generate you one automatically so you can copy it as a starting point, because it has some header information.

Files of interest

1. **activity_main.xml** This uses a ConstraintLayout. We even preserve the centered text box.

   The design view helps when working with constraint layouts. You will want to look at chains, which you can define with the GUI and modify the XML text. You will not need any nested layouts.

2. **activity_main_linear.xml** Use only LinearLayouts, including nested LinearLayouts. Remember, layout_weight is your friend, as is gravity (position within an object (e.g., a button within a LinearLayout)), and layout_gravity (position of an object in its parent).

   The nesting on the bottom layout gets a little deep. That makes it tricky, so if you are getting too frustrated, just add an invisible button (android:visibility="invisible"). The invisible button solution will get you most of the points, but you can do it without an invisible button for all the points. For full points you can use an empty linear layout!

No code needs to execute, so you are off the hook there!