COSC 3P97 Assignment 1

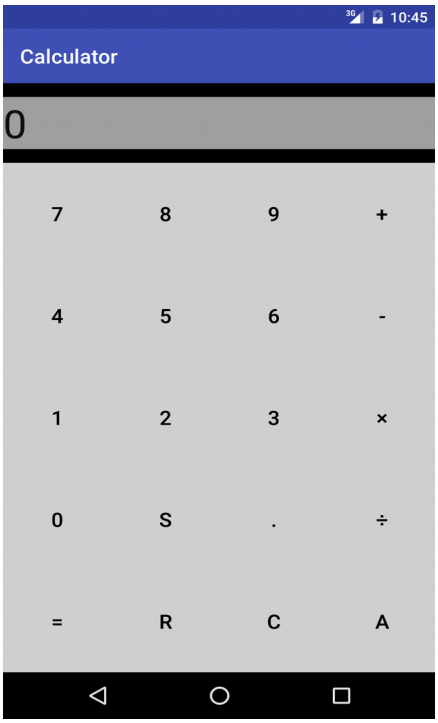
Fall 2018/19

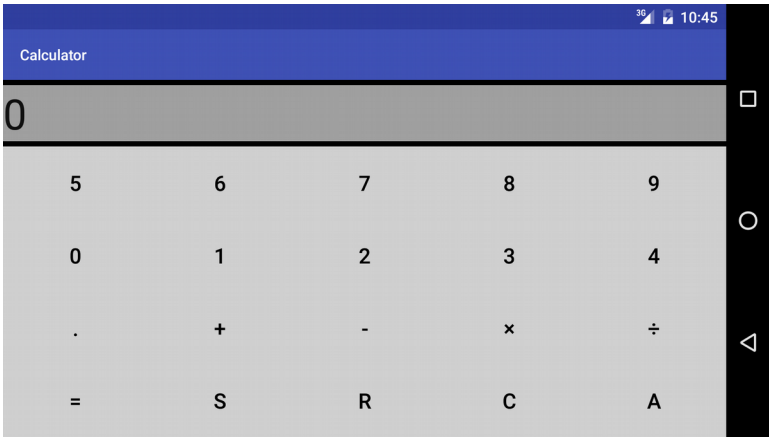
Due: Oct. 12 @ 5:00 pm.

Create a new Android Studio project or Eclipse workspace for the assignment. The app should run on API 23 (Marshmallow).

Calculator

Write an Android app to provide a simple calculator; base example shown below.





Basic operation is simple, but you'll be required to choose 3 of the 6 additional features listed below to add as well.

Note: the included screenshots are not necessarily indicative of how your submission should **look**. It's just a visual aid. Feel free to create a better design, if you like.

Basic Operation:

This will earn you the vast majority of the marks, come evaluation.

- Employ a layout that arranges the display and buttons such that they make reasonably effective use of the screen space
- Create separate layouts for portrait and landscape views
- Press a number key to append to the current operand
    - If the display has just been *cleared*, or an operator was the last key pressed, this starts creating a new operand
- Press an operator key to select the operation
    - In this basic mode, pressing two operators in a row may either have the second replace the first, or perform the first operator (using the first operand as both operands). e.g. 3*+ could be either be replaced as 3+, or 9+, depending on your preference
    - In this basic mode, pressing an operator after operand→operator→operand will first act as though you pressed the = key, and then pressed the selected operator, with the result of the prior calculation being the first operand
    - Similarly, an = key will either perform a null operation if a single operand has been entered (e.g. 23= simply yields 23), or will perform the operation if the second operand has been provided (e.g. 2+3= yields 5)
- Press your correct button (e.g. C) to either remove the last operator or the last digit (depending on which was pressed more recently)
    - If it acts on an operand, it's acceptable to clear the entire operand
    - If the last operation was an =, or to start with a clear display, there's no expected behaviour
- Press the All-Clear button (A, AC, Clear, etc.) to restore the display to 0 or blank, with no operators/operands selected
- If an operation is invalid (e.g. division by zero), either display an error, or reset the display to 0 or -1 (per your preference)

Additional Features:

For full credit, you'll also need to make any **three** of the following **six** additions:

1. Allow for parentheses — ( )
    - Note: If you choose this, you need to be able to *nest* them
2. Add and implement the decimal (.) key
    - Pressing the decimal key a second time during the same operand should either be ignored, or move the decimal to the current position, per your preference
    - Pressing the decimal key after finishing an operation could either append a decimal after the last result, be ignored, or start a new operand with 0.
3. Interpretation of the - key as a *negative* sign in operations
    - This mostly only makes sense as a choice if you also go with #5 below
4. Add and implement a *memory* feature
    - e.g. M+ or MS and MR (S and R in the picture above)
    - When you press to store, the operand/current working value is placed into memory. When you press to recall, the stored value becomes the current operand/working value
    - It's up to you whether you should be able to add additional digits to the recalled value
    - Operators are not stored in memory
    - If a new value is stored, it may simply replace the old one
    - The default stored value (before you press Store) is 0
    - There's no requirement for persistence. A value need only be remembered within the current execution of the application
5. Formula entry
    - Rather than only displaying a single operand at a time, queue in an entire expression, displaying as it's entered, to be calculated once = is pressed
    - This must follow standard order of operations for operators (including parentheses, if present)
6. If you do #5, a radio button (or comparable) for switching between formula entry and 'basic' mode

Note that some of the above are easier than others. However, they're all worth the same amount as each other; no selection of 3 is worth any more than another.

Tips:

- Be careful how your program behaves when, for example, rotating the display. A calculator is not an etch-a-sketch; you're allowed to turn it
- Please note the requirement for using Marshmallow. If you don't, the marker may not be able to run your submission, or some things may not appear the way you expect; if you don't follow instructions, you'll be graded on how they appear for the marker
- Calculator displays aren't directly editable
- If you do choose to use GridLayout, note that (as of the more recent SDKs, including 23/Marshmallow) you can use weights
    - But that in no way means GridLayout is mandatory
- The screenshot above really is just a visual aid. You may find it easier to, for example, have some buttons span multiple button widths/heights, or add additional spacers, etc.
- Please remember that documentation still matters
    - So does *reasonable* abstraction and decomposition of functionality (e.g. methods, etc.)
- There should be no sequence of button presses that can crash your program

Submission

Your submission will be only electronic. You should have an Android Studio project for the assignment on your `Z:` drive.

You might want to delete the 'build' and '**.**gradle' folders to trim the size.

Important: In order to receive any credit, you'll also need to include a built APK. Just click Build, followed by Build APK(s). When it's done, it'll give a notification in the lower-right. Click 'Locate' to see the folder (it'll just be the build/outputs/apk/debug path of your project). Move that .apk file higher up to be beside your project (where the marker can easily find it), and ideally rename it to something more descriptive.
Your submission will consist of both your project *and* that apk!
(No apk means you get a zero)

Note: If you don't actually tell it to build the apk, and just take whatever it autogenerated, that won't work on other devices/simulators!

Log on to sandcastle and navigate to the project or workspace directory, at the sandcastle prompt, type:

```
submit3p97
```

and, when prompted, enter the appropriate information using `1` as the assignment number. The submission program will copy the entire working directory, and all subdirectories, to the marking directory, so you should ensure that the content of the directory is what you desire to submit.

If you want to make grading as easy as possible for the marker, and expect feedback, then also consider including a .pdf of sample output, instructions, or anything else you feel pertinent.