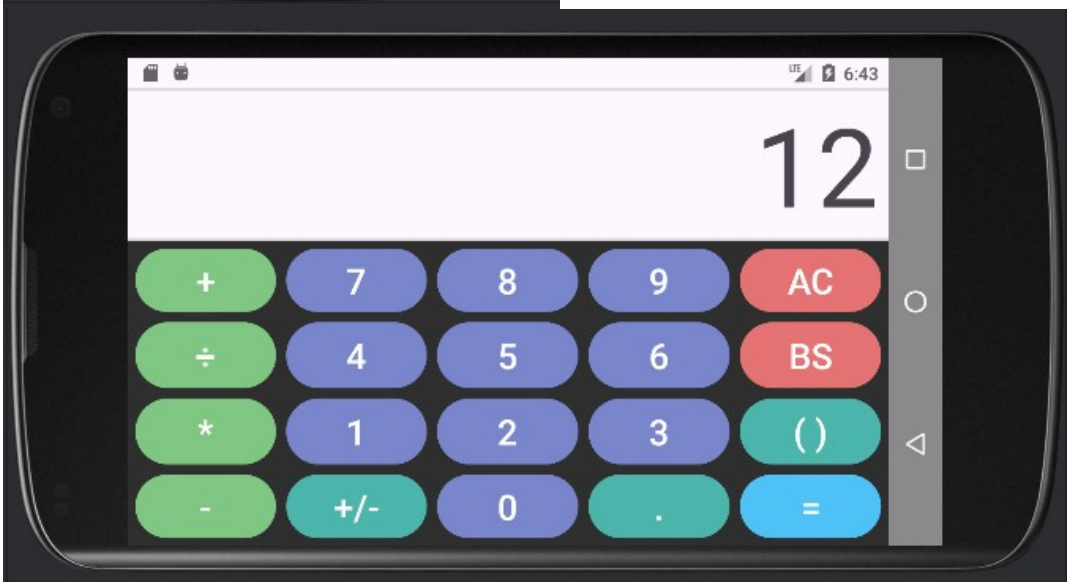
Brett Terpstra
6920201
bt19ex@brocku.ca
COSC 3P97 Assignment 1



The code used for this assignment is fairly simple, it features a simple LR parser similar to the one Ali had us make for 2P05, except this one actually produces a working output. Yes, this means that I decided to implement formula entry.

The user interface was designed with usability in mind, the landscape features easy access to the operator buttons as if you are using the calculator in landscape you are likely to be using your phone with two hands.

The AC button will clear the entire formula entry while the BS button will remove a single entry from the input. Note entry here is considered a character.

Additional Features:
1. Parenthesis

This app features formula evaltion so parenthesis were a logical choice. There is a single button to add them and the app will automatically determine which one is the most appropriate given your input (either ‘(‘ or ‘)’). I considered making it wrap single numbers (by inserting a ‘(‘ before them) instead of assuming multiplication but as this was not a requirement and I already added the other feature, this was not added. If a single opening brace is the last entered it will be treated as an operator and replaced if you try to type a different operator. This is a nice QOL feature

Example formula, which evaluates correctly (showing order is respected.):

2. Decimal numbers

This is pretty simple and explains itself. If there is already a decimal in the number it does nothing. If a partial decimal is enter the calculator will remove the unused decimal place. Eg: 1. -> 1

3. Negative Numbers

This is also pretty simple but comes with a few nice features. If you press the +/- key it will convert the current expression into either the negative version or back to the positive version. Expressions can either be contained within parenthesis or single numbers. If the last entry in the formula is a closing parenthesis, the entire thing will be negated while if it is an open number that number will be negated. If the number is already negative it will be converted to a positive number.

Extra case includes if there is a negative sign already present, another will be added; allowing for:
Example chains of pressing the button

```
4 - -2   -> 4 - 2        -> 4 - -2       -> ...
4 + 2    -> 4 + -2       -> 4 + 2        -> ...
(4 + 2   -> (4 + -2      -> (4 + -2      -> ...
(4 + 2) -> -(4 + 2)      -> (4 + 2)      -> ...
```

and so on.

4. Formula Entry:

As the wording of this requirement implies that this is a mutally exclusive option only formula entry was implemented. No basic mode is present. (plus #6 is an optional feature to add a switch so that only add extra reason to this it is optional.) And foxwell said in class that it is fine to not implement basic mode if you have formula entry. Anyways formula entry follows standard BEDMAS. As there is no exponent BDMAS is followed. As mentioned earlier the string input is tokenized then parsed using an LR parser encoding the presidence levels as states in the EBNF.

Rotating the app keeps the display data. Dividing by zero displays an error. Everything is dynamically scaled and layouts are different between landscape and portrit (and making effective use of the screen).

This should be all the features I have added, there might be a few edge case (not required) QOL features I have added which are slipping my mind. I belive this marks all requirements for the assignment, plus one extra extra feature was added. The Java and XML are documented where I felt necessary and code makes use of appropriate abstractions. I hope you enjoy the look of my calculator, that all your check boxes are ticked, and that this document has made it easier to mark.

Have a good day and happy Halloween

:3