CSCI 457 Assignment 2 – Hexadecimal Calculator

Implement a hexadecimal calculator for iOS.

Requirements:

- The calculator should support 4 basic arithmetic operations: + - * and /
- The calculator will operate on hexadecimal numbers, not decimal numbers
- The calculator only needs to operate on unsigned integers (i.e. UInt). You do not need to consider negative numbers or fractions.
- The calculator should support the 16-digit hexadecimal numbers (i.e. The range of the numbers is from 0 to FFFF FFFF FFFF FFFF). Prevent the user from entering a number that is greater than FFFF FFFF FFFF FFFF.
- The calculator should handle overflow and underflow gracefully. The app must not crash.
- The calculator should handle division-by-zero error gracefully. The app must not crash.
- The calculator should be able to support most of the devices and orientations. If it does not support the old devices earlier than iPhone 6, it is okay.

Hint:

- To convert a string to a hex number, use "radix: 16" as an argument. For example:

```
var s:String?
s = "1A"
var intHex:UInt = 0

intHex = UInt(s!, radix: 16)!
print(intHex)    // shows 26

intHex = 90
s = String(intHex, radix: 16).uppercased()
print(s!)        // shows 5A
```

- It is recommended that you use a UI label instead of a text field, so that the user will not type directly by using a keyboard. You will need to provide a button for each digit.
- Strings may be concatenated by using + operator. E.g.

```
var s1 = "1234"
var s2 = "5"
print(s1 + s2)   // shows 12345
```

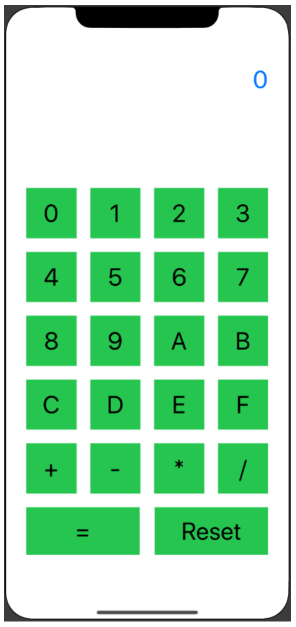
You may want to do string concatenation in the action of each digit button.

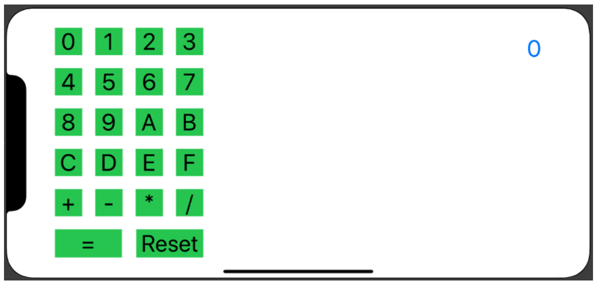
- To prevent the user from entering a number exceeding the size of 16 digits, you may verify the length of the string associated with the UI label.
- To handle overflow and underflow, use &+, &-, and &* instead of +, -, and *.
- To support different devices and orientations, use stack view, scroll view, or both.
- Design your algorithm first! Think about the status of the calculator: when to take the first operand, when to take the second operand, when to append digits to the current number, and when to refresh the current number, etc.

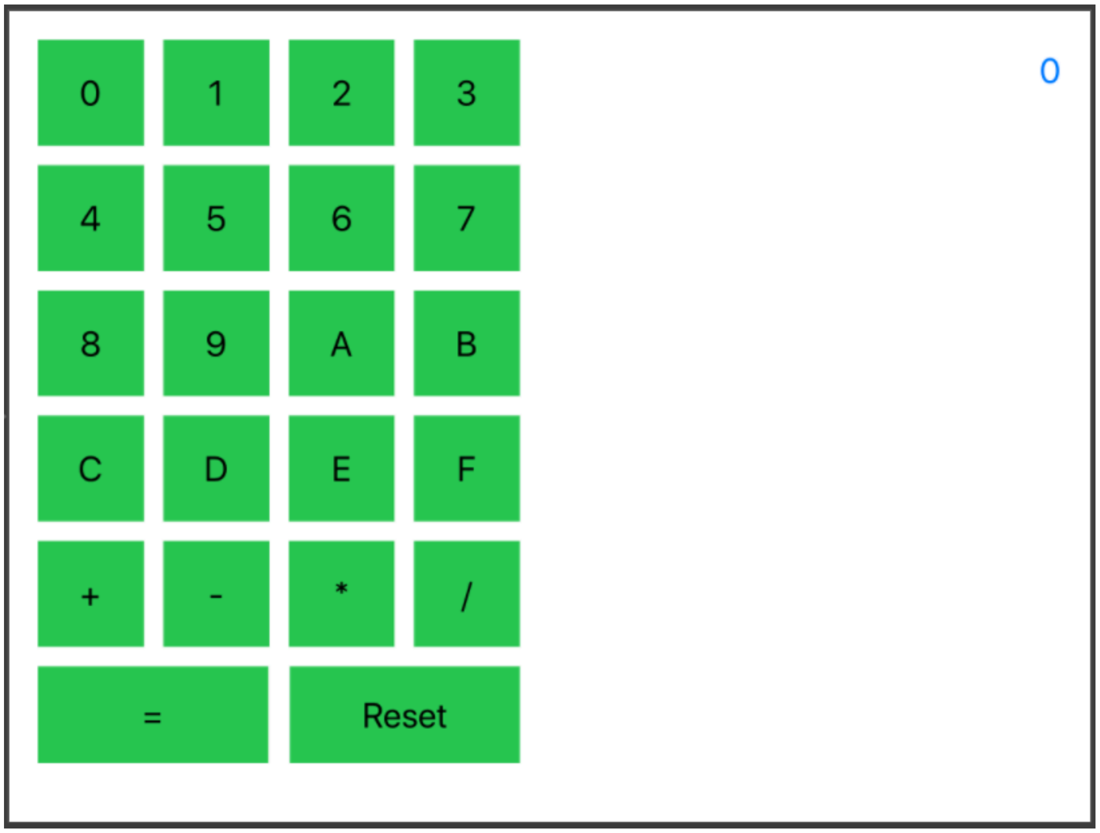
The functionality of your hex calculator worth 60% of the credit, while the appearance of the user interface worth 40%.

When you submit the assignment, please compress the entire project folder into a single zip file, and upload it to D2L. In addition, please provide 4 to 5 screenshots of your app in different devices and orientations. If your app doesn’t work on every device/orientation, please specify why.

The screenshots of a sample program are shown below. Your UI does not have to be the same as the sample program. As long as it has a pleasing looking, it should be fine.






0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
+
-
*
/
=
Reset
0