```
import Foundation

/*
  a simple class implementing a base-conversion calculator
  supports bases 1-16, using hex standard a-f for bases 11-16.
*/
class Bases {

    // our instance variable, with default value
    private var input:String = ""

    // constructor "self" is sort of like "this" in Java
    init(input:String) {
        self.input = input
    }

    private func toLower(b:Int) -> String {
        // precondition
        if(b < 0 || b > 10) {
            return ""
        }

        // input might not be coercable to an Int
        // so if it isn't, we use the value nil
        // summary: option types are for safety, sanity
        var digits = Int(input) ?? 0

        var retval = ""
        repeat {//loop
            let mod = digits % b
            digits = digits / b
            retval = String(mod) + retval
        } while(digits != 0)
        return retval
    }

    private func toUnary() -> String {
        let limit = Int(input) ?? 0
        return String(count: limit, repeatedValue: Character("0"))
    }


    private func toUptoHex(b:Int) -> String {
        if(b <= 10 || b > 16) {
            return ""
        }

        var digits = Int(input) ?? 0
        var retval = ""
        repeat {
            let mod = digits % b
            digits = digits / b
            // like C switch, but no fallthrough
            // also must be assign/return/etc., not just a value
            switch mod {
            case 15: retval = "f" + retval
```

```
            case 14: retval = "e" + retval
            case 13: retval = "d" + retval
            case 12: retval = "c" + retval
            case 11: retval = "b" + retval
            case 10: retval = "a" + retval
            default: retval = String(mod) + retval
            }
        } while(digits != 0)
        return retval
    }

    func convert(base:Int) -> String {
        if(base == 1) {
            return toUnary()
        } else if (base <= 10) {
            return toLower(base)
        } else if (base <= 16) {
            return toUptoHex(base)
        } else {
            return ""
        }
    }
}


// test client just hanging out outside class b/c playground
let str = "12"
let bc = Bases(input:str)
let binary = bc.convert(2) // bc.toLower(2)
let octal = bc.convert(8) // bc.toLower(8)
let decimal = bc.convert(10) // bc.toLower(10) // silly
let duodecimal = bc.convert(12) // bc.toUptoHex(12)
let hex = bc.convert(16) // bc.toUptoHex(16)
let unary = bc.convert(1) // bc.toUnary()
String(unary).characters.count
```