Programming Assignment 2

Game of Sticks (VIM)
Modified from: http://nifty.stanford.edu/2014/laaksonen-vihavainen-game-of-sticks/handout.html

Assigned: Wednesday, September 14, 2016
Due: Wednesday, September 28, 2016 at midnight

Game of Sticks

In the game of sticks there is a heap of sticks on a board. On their turn, each player picks up 1 to 3 sticks. The one who has to pick the final stick will be the loser.

The following is an example of the game of sticks.

- The game starts with 20 sticks on the board.
- Marvin takes 3 sticks, there are 17 sticks remaining.
- CPU takes 2 sticks, there are 15 sticks remaining.
- Marvin takes 1 stick, there are 14 sticks remaining.
- CPU takes 3 sticks, there are 11 sticks remaining.
- Marvin takes 2 sticks, there are 9 sticks remaining.
- CPU takes 2 sticks, there are 7 sticks remaining.
- Marvin takes 3 sticks, there are 4 sticks remaining.
- CPU takes 1 stick, there are 3 sticks remaining.
- Marvin takes 2 sticks, there is 1 stick remaining.
- CPU has to take the final stick and loses.

Part one: Human vs. Computer

Modify assignment 1 to where a player can play against the computer. Initially, the computer will pick up sticks from 1..3 selected randomly, but if the player continues playing another game then you will help the computer learn and improve by using results from each game.

In order to improve the choices selected by the computer, you will integrate stick-choices when the computer wins a game, and remove them when the computer loses. You start out with a Dictionary, for 1..100 starting sticks. Therefore, at the end of each game ask the user if they want to continue or quit, if they choose to continue then you will update the Dictionary to increase the computer's likelihood of winning . Use a Dictionary for storing the winning values.

This is the initial state of the Dictionary:

Amount of Sticks	Sticks the CPU can take
1	1
2	1,2
3	1,2,3
4	1,2,3
5	1,2,3
6	1,2,3
7	1,2,3
8	1,2,3
9	1,2,3
10	1,2,3

When it is the computer's turn, you randomly select a stick to be removed. For example, if there were 7 sticks left then you would great a random index between 0 and 2, return the value at the index. This is hard to see right off because there is a 1:1 ordering between the index and sticks in the set.

Let's look at the Dictionary later in the game. An example of what the Dictionary might look like after a few games:

Amount of Sticks	Sticks the CPU can take
1	1

2	1,2,1,1,2
3	1,2,3,1,3,2,1,2,3
4	1,2,3,2,2,1,3,1
5	1,2,3,3,2,2,1
6	1,2,3,1,1,1,1
7	1,2,3,2,2,2,1,1
8	1,2,3,2,3,2,2
9	1,2,3,2,1,2,2
10	1,2,3,2,1,3,2

In this

You will want to save the Computer moves to the Dictionary when it wins, building up the probability of the computer selecting a winning number of sticks each time. If it loses, remove the moves it took from the Dictionary, lowering the probability of selecting a number of sticks that results in a loss. However, do not remove an option all together

For example of a loss by the computer, assume in this game we started with 10 sticks. The computed selected sticks three times, and there were 8, 3, & 1 sticks left, respectively. The number of sticks removed by the computer were 3, 1, 1 respectively. To train the computer, you remove 3 for Dictionary value for 8, and a 1 for 3. If you look at the table below you will notice that on row 8 & 3 the sticks the CPU can take has had the 3 & 1 removed from the row. Row 1 still has 1 because there was not multiple 1's in the that row.

Amount of Sticks	Sticks the CPU can take
1	1
2	1,2,1,2
3	1,2,3,1,3,2,2,3
4	1,2,3,2,2,1,3,1
5	1,2,3,3,2,2,1
6	1,2,3,1,1,1,1
7	1,2,3,2,2,2,1,1

8	1,2,3,2,2,2
9	1,2,3,2,1,2,2
10	1,2,3,2,1,3,2

PART TWO BONUS:

The bonus is to train the computer to select winning stick choices over time, not just within 1 game play.   The problem with part is each time the user quits then s/he restarts with an ‘ignorant’ computer.

For the bonus, you will save the moves taken to a txt file at the end of each game and read the moves in at the start of the game so the computer can reuse the accumulated moves. Use the 2 functions provided below for writing to the txt file and also for reading in the txt file.

```
func write(file:String,joined:String) {
        if let dir = NSSearchPathForDirectoriesInDomains(NSSearchPathDirectory.DocumentDirectory,
NSSearchPathDomainMask.AllDomainsMask, true).first {
            let path = NSURL(fileURLWithPath: dir).URLByAppendingPathComponent(file)

            //writing
            do {
                try joined.writeToURL(path, atomically: false, encoding: NSUTF8StringEncoding)
            }
            catch {/* error handling here */}
        }
    }

  func read(file:String) -> String {
        var txt = ""
        if let dir = NSSearchPathForDirectoriesInDomains(NSSearchPathDirectory.DocumentDirectory,
NSSearchPathDomainMask.AllDomainsMask, true).first {
            let path = NSURL(fileURLWithPath: dir).URLByAppendingPathComponent(file)

            //reading
            do {
                txt = try NSString(contentsOfURL: path, encoding: NSUTF8StringEncoding) as String!
            }
            catch {/* error handling here */}
        }
        return txt
    }
```

Example 1

```
Welcome to the game of sticks!
How many sticks are there on the table initially (10 - 100)?
15
There are 15 sticks on the board
Player 1: How many sticks do you take (1 - 3)?
```

```
2
There are 13 sticks on the board
There are 11 sticks on the board
Player 1: How many sticks do you take (1 - 3)?
1
There are 10 sticks on the board
There are 7 sticks on the board
Player 1: How many sticks do you take (1 - 3)?
2
There are 5 sticks on the board
There are 4 sticks on the board
Player 1: How many sticks do you take (1 - 3)?
3
There are 1 sticks on the board
CPU, lost.
```

Example 2

```
Welcome to the game of sticks!
How many sticks are there on the table initially (10 - 100)?
10
There are 10 sticks on the board
Player 1: How many sticks do you take (1 - 3)?
1
There are 9 sticks on the board
There are 6 sticks on the board
Player 1: How many sticks do you take (1 - 3)?
1
There are 5 sticks on the board
There are 2 sticks on the board
Player 1: How many sticks do you take (1 - 2)?
2
Player 1, you lose.
```

Implement a game with the functionality described above in Swift using functions, dictionaries, optionals and user input.

Grading Rubric:

Items	**Percent age**	**Expert (100)**	**Proficient (75)**	**Needs Improvement (50)**	**Unsatisfactory (0)**

Input and output match the text above	10%	Matches 100%	Up to 2 typos	More than 2 errors but fewer than 5	More than 5 different errors in input/output
Use of functions	20%	Functions are cleanly defined (do X)	Use of functions, but do not add to readability or understanding of code	Use of functions results in code being harder to understand	No functions used
Plays game correctly	40%	Plays games for number of sticks between 10 and 100, inclusive	Generally works, but misses ‘special’ or edge cases	Game can’t be played as described	Game doesn’t compile
Handles erroneous input correctly	20%	Both cases (total sticks and ones picked up)	Misses one of the 4		Misses more than one of the 4 cases
Adheres to CSC 416 coding guidelines for Swift	10%	Miss none of the style code	Misses 3 or fewer coding rules (may be more unique errors)	More than 3 different type of coding style errors, but fewer than 10	More than 7 different violations of Swift programming style