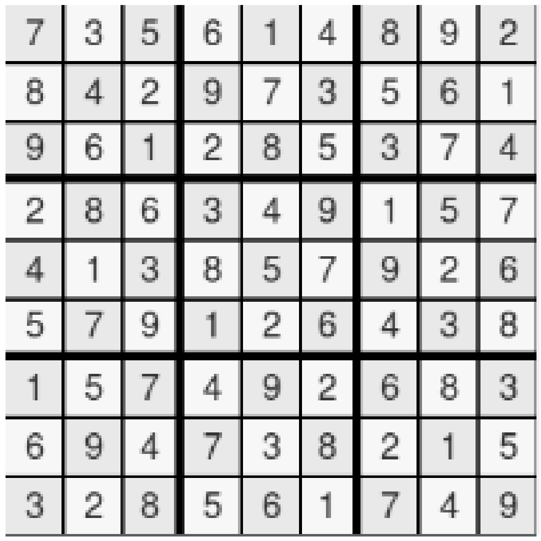
CS10 In-Lab Midterm - Summer 2019

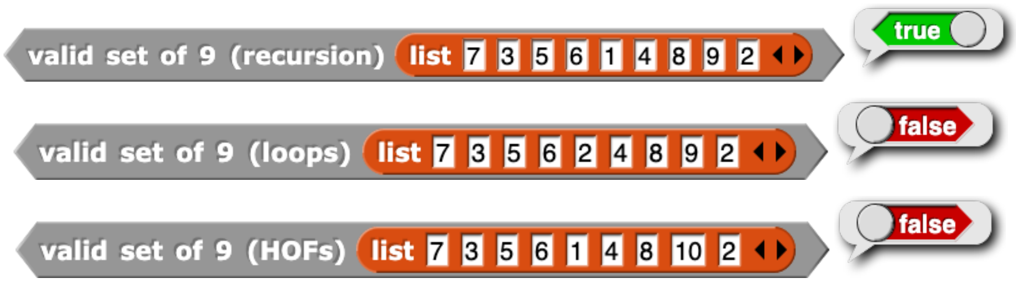
A Sudoku is a number puzzle: to solve it, each row, column, and each smaller square (bolded below) has to consist of the numbers 1-9. Below is an example of a solved Sudoku:



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

We want to build a block that takes in a Sudoku and determines if it was filled in correctly. Follow the instructions below to create this block. Use the starter file here (bjc.link/sd82j9); you do not need to create a new Snap! file.

1. Create a helper block that takes in a list of 9 numbers and determines if they would form a valid Sudoku row/column/square (as a reminder, a valid Sudoku row/column/square has each of the numbers from 1-9 appear once). There are options for you to write this block using loops, higher order functions, and recursion. You are only required to write it two ways; if you write it all three ways you will get 3 bonus points. Below are some sample calls to the block:



```
valid set of 9 (recursion) (list 7 3 5 6 1 4 8 9 2)   → true
valid set of 9 (loops) (list 7 3 5 6 2 4 8 9 2)       → false
valid set of 9 (HOFs) (list 7 3 5 6 1 4 8 10 2)       → false
```

2. Fill in the skeleton code for "valid Sudoku" so that it correctly reports whether or not a Sudoku is valid. We have written the following helper blocks to help you with this part of the exam:
   - `item ( ) , ( ) of [ ]` takes in a row, column, and a Sudoku and outputs the item at that row, column of the Sudoku
   - `find squares [ ]` takes in a Sudoku and outputs a list of lists, where each sublist is a 1D representation of a 3x3 square on the board.