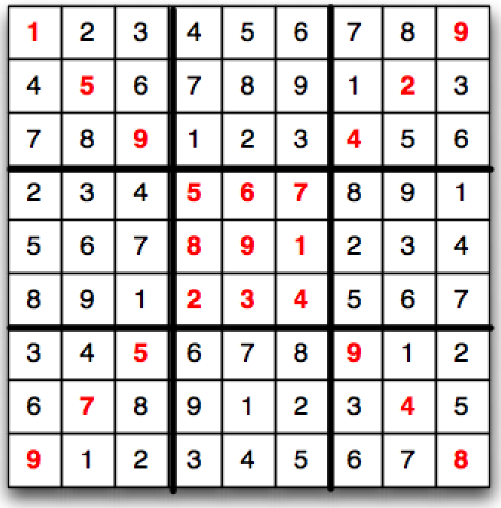
Functional Programming Skills
Assignment 2

Arthur Nunes-Harwitt

Each question is worth 10 points.
Explicitly write the type of each function.



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

Sudoku is a puzzle, involving a grid of 9 rows and 9 columns, which is subdivided as a grid of 3 times 3 boxes with 3 rows of 3 columns each. Each of the 81 cells must contain a digit from 1 to 9, so that each row, column, and box contains all nine digits. A puzzle instance has some squares with no digits. Solving the puzzle means filling in the blank squares so that the constraints are satisfied. You will write a sudoku solver using the generic solver developed in class.

You will need to do the following.

1. Create a data type called `SudokuConfig`.
2. Write a function `sudokuConfigFromList` that takes a list of integers and returns a `SudokuConfig`. Zero represents a blank.
3. Write a function `listFromSudokuConfig` that takes a `SudokuConfig` and returns a list of integers. Again, zero represents a blank.
4. Make `SudokuConfig` an instance of `Eq`.

5. Make `SudokuConfig` an instance of `Show`. The output generated should correspond to the example below. In the file `Problems.hs`, `trivial` is defined as follows.

```
-- from Page-A-Day Sudoku Calendar, April-19-2008
trivial =    [ 0, 4, 6,  0, 0, 0,  8, 9, 0,
               0, 7, 0,  4, 0, 9,  0, 1, 0,
               5, 0, 0,  0, 8, 0,  0, 0, 6,

               0, 0, 3,  9, 0, 8,  6, 0, 0,
               9, 0, 0,  0, 0, 0,  0, 0, 2,
               0, 0, 8,  5, 0, 2,  1, 0, 0,

               4, 0, 0,  0, 5, 0,  0, 0, 3,
               0, 2, 0,  1, 0, 6,  0, 7, 0,
               0, 9, 7,  0, 0, 0,  5, 2, 0 ];
```

   Importing that definition leads to the following interaction.

```
*Assign2> Just (sudokuConfigFromList trivial)
Just
_  4  6    _  _  _    8  9  _
_  7  _    4  _  9    _  1  _
5  _  _    _  8  _    _  _  6

_  _  3    9  _  8    6  _  _
9  _  _    _  _  _    _  _  2
_  _  8    5  _  2    1  _  _

4  _  _    _  5  _    _  _  3
_  2  _    1  _  6    _  7  _
_  9  7    _  _  _    5  2  _
*Assign2>
```

6. Make `SudokuConfig` an instance of `Config` by defining the `successors` function and the `isGoal` function.

7. Write a function `sudokuSolve` that takes a `SudokuConfig` and returns a `Maybe SudokuConfig`.

Graduate Problem/Undergraduate Extra Credit

The satisfiability problem is the problem of determining, given a Boolean formula φ, whether or not φ is satisfiable (i.e., whether there is an association of the variables in φ to truth values such that φ evaluates to true). Boolean formulas can be characterized recursively as follows. You will write a satisfiability solver using the generic solver developed in class.

A Boolean formula is one of the following.

- a Boolean constant b
- a variable x
- $(\varphi_1 \wedge \varphi_2)$, where φ_1 and φ_2 are Boolean formulas
- $(\varphi_1 \vee \varphi_2)$, where φ_1 and φ_2 are Boolean formulas
- $(\neg\varphi)$, where φ is a Boolean formula

You will need to do the following.

1. Create a data type called `BExp` to represent Boolean formulas. It should derive `Eq` and `Show`. The constructors should be called `BConst`, `Var`, `And`, `Or`, and `Not`. Variable names are expressed using strings.
2. Create a data type called `SatConfig`. It should be an instance of `Eq`. Its associated `show` function should display variables and their associated truth values. Make `SatConfig` an instance of `Config` by defining the `successors` function and the `isGoal` function. The function `successors` should prune away successors that can't possibly lead to a solution.
3. Write a function `satSolve` that takes a formula of type `BExp` and returns a `Maybe SatConfig`.