Sudoku Foundation Classes
v1.4
Cpt S 489

Read all instructions carefully (6 pages) before writing any code. Do all work individually!

The zip that contains these instructions also includes an HTML and JavaScript code file. The HTML page is set up to reference the JavaScript code file and use it to run tests. Do not change anything in this file. The JavaScript file is where you must write the code.

In this assignment, you will implement a set of classes relating to solving a Sudoku puzzle. The actual solving logic will be implemented in the next assignment. This assignment is just about creating the following classes (constructor functions), which will provide a foundation on top of which a solver can be built:

- **Sudoku9x9** - Represents a full 9x9 Sudoku puzzle. Stores an array of 81 SudokuCells objects.
- **SudokuCell** - Represents a single, mutable cell in a Sudoku puzzle. Stores a list of numbers representing possible values for the cell. Should this list have 1 item, the cell has a known, finalized value.
- **SudokuCellCollection** – Represents an immutable collection of SudokuCell objects.
- **SudokuCellBlock** - Represents a collection of cells that is either an entire row, an entire column, or a 3x3 box. In other words, a collection of 9 cells that must have values [1,9] among them when solved. Inherits from SudokuCellCollection.
- **Sudoku3x3Block** – Represents one of the nine 3x3 box areas from a Sudoku9x9 puzzle. Inherits from SudokuCellBlock.

First familiarize yourself with the rules of classic Sudoku. Do a few practice puzzles from newspapers, online apps, iOS or Android apps, or elsewhere. Although it's more relevant for the next assignment, you may also want to read some about the strategies for solving the puzzles. At a minimum, be familiar with the idea of having each cell containing a list of possible values and removing items from that list based on logic rules until only one item remains in the cell. This is the typical logic-based approach to solving a Sudoku puzzle.

Implement the members for each class according to the specifications that follow.

Sudoku9x9 Class	
Member	**Description**
Sudoku9x9 (constructor function)	function Sudoku9x9(arrOf81Values) Constructor function that initializes the member array of 81 cells. Assume that the arrOf81Values parameter is guaranteed to be an array with exactly 81 items. Each item in arrOf81Values is either a numerical value in the range [1,9] or anything else. If it is a number in the range [1,9], this indicates that that cell has a known, finalized value. If it is anything else, that means it represents a "blank" cell, with all values in the range [1,9] being possible. Initialize the Cells member array appropriately based on this input.
get3x3	function(rowIndex, colIndex) Function that returns a Sudoku3x3Block for one of the 9 possible 3x3 boxes in the puzzle. The row and column indices will be in the range [0,2].
getColumn	function(colIndex) Function that returns a SudokuCellBlock containing the 9 cells from the specified column. The column index will be in the range [0,8].
getRow	function(rowIndex) Function that returns a SudokuCellBlock containing the 9 cells from the specified row. The row index will be in the range [0,8].
toArray	function() Function that returns a deep copy of the array of cell references.

SudokuCell Class	
Member	**Description**
SudokuCell (constructor function)	function SudokuCell(numPossibleValues) Constructor function for a SudokuCell. You may assume that numValues is always equal to 9. It exists only to allow for flexibility to support other types of Sudoku puzzles that have potentially more or fewer possible values per cell. (Ex: There are large 16x16 Sudoku puzzles that support 16 possible values per cell). Initialize the cell such that all values in the range [1,9] are listed as possibilities for this cell.
containsPossibility	function(value) Returns true if the cell contains the specified value as a possibility, false otherwise.
finalizedValue	Read/write numerical value property that represents the finalized value for the cell. When getting the value, if isFinalized is false, then the return value is undefined. Otherwise the finalized value for the cell is returned. When setting, all other possibilities are removed from the cell and it becomes a cell with just one possible value.
getPossibilities	function() Gets a sorted array of numbers representing possible values for this cell.
isFinalized	Read-only bool property that is true only if the cell has been narrowed down to 1 possible value. When the cell is in the finalized state it becomes immutable.
removePossibility	function(value) Removes a value from the list of possible values for this cell. This is OK to call even if the value is not currently in the list of possible values for this cell. If the cell is finalized, then no action is taken, even if the value is equal to the finalized value in the cell. Returns true if the value WAS a possibility in this cell and now it is not. Returns false in all other cases.
removePossibilities	function(arrOfValues) Utility function that calls removePossibility for each value in arrOfValues.
toString	function() Returns the string version of the array built by getPossibilities.

SudokuCellCollection Class	
Member	**Description**
SudokuCellCollection (constructor function)	function SudokuCellCollection(arrOfCells) Constructs an immutable collection of cells. Here "immutable" means that no cells can be added to or removed from the collection after instantiation. However, note that it is a collection of SudokuCell objects, and each such object is mutable. Makes a deep copy of the passed array.
containsCell	function(cell) Does a reference search for the specified cell. Returns true if it is contained within this collection, false if it is not.
containsPossibility	function(value) Loops through all cells in this collection and returns true if any one of them contains the specified value as a possibility.
count	function(predicate) Counts the number of cells that satisfy the specified predicate. Calls the predicate function on each cell, and returns the number of times true is returned. Ex: If you had a cell collection named 'coll' you could count the number of finalized cells in this collection with the following statement: coll.count(function(cell) { return cell.isFinalized; })
forEach	function(functionThatTakes1CellParam[, startIndex]) Goes through cells in this collection and runs the functionThatTakes1CellParam function on each one. Starts at the specified starting index if it is provided, otherwise starts at index 0.
getFinalizedValues	function() Builds and returns array of numerical values, one for each finalized cell value in the collection. Returns an empty array if no cells in the collection are finalized.
getPossibilities	function() Builds an array of distinct numerical values that represents all possible values among all non-finalized cells. The returned array is sorted in ascending order.
length	Read-only numerical property that returns the number of cells in the collection.
removeCell	function(cell) Returns a new cell collection that has all the cells that this collection has, minus the specified cell. (Remember that the cell collection is immutable, hence the need for it to build and return a new collection)
removeCells	function(otherCellCollection)

	Builds and returns the collection of cells that are only in this collection and not in the other. If the otherCellCollection parameter is not a SudokuCellCollection object, then a reference to this collection is returned.
removePossibility	function(value) Removes the value as a possibility from all cells in the collection. Returns the number of cells that had the value as a possibility and had it removed.

SudokuCellBlock Class	
Member	**Description**
SudokuCellBlock (constructor function)	function SudokuCellBlock(arrOf9Cells) Constructor function for a collection of cells that is either an entire row, an entire column, or a 3x3 box. (Reminder: SudokuCellBlock inherits from SudokuCellCollection)
trySolve	function() Tries to solve this block. This function looks only at the values in the collection and doesn't involve the other 8/9 of the puzzle. This is a small piece of the logic needed to solve an entire puzzle. An object is returned that gives information about the attempt at solving. This object has 2 properties: "changed" and "solved", both of which are bool values. The "changed" property is set to true if any changes were made to any of the cells in the collection. This includes even just eliminating one possibility from one cell, and leaving it unsolved with multiple possibilities left. If the state of any cell is changed due to this call to trySolve, the "changed" value must be true. If no changes were made, then "changed" must be false. The "solved" property is set to true only if all cells in the collection have finalized values (meaning that this block is fully solved).

Sudoku3x3Block Class	
Member	**Description**
Sudoku3x3Block (constructor function)	function Sudoku3x3Block(arrOf9Cells) Constructs the 3x3 cell collection from a single-dimensional array of 9 cells. The first 3 cells in arrOf9Cells are the first row of the 3x3 block, the next 3 the second row, and the last 3 the bottom row. Reminder: Sudoku3x3CellColection inherits from SudokuCellBlock.
getPossibilitiesOnlyAvailableOnColumn	function(collIndex) Gets the possibilities within the 3x3 block that are unique to the specified column within the block. No values in the returned array will be possibilities for cells that aren't in the specified column within this 3x3 block. The column index must be in the range [0,2].
getPossibilitiesOnlyAvailableOnRow	function(rowIndex) Gets the possibilities within the 3x3 block that are unique to the specified row within the block. No values in the returned array will be possibilities for cells that aren't in the specified row within this 3x3 block. The row index must be in the range [0,2].
isColumnFinalized	function(columnIndex) Function that returns true if all cells in the specified column are finalized, false otherwise.
isRowFinalized	function(rowIndex) Function that returns true if all cells in the specified row are finalized, false otherwise.
toString	function() (optional, may help with debugging) Function that returns a string representing the contents of this 3x3 block. Example of format you might want: “[4, 5, 1] [9, 2, 6] [7, 6, 8]”

You may add additional members to each class as needed, provided you don’t violate any rules of the class. For example, if you add extra members to SudokuCellCollection or inheriting classes, the collections must remain immutable.

Scoring:
All tests pass: 3/3
All tests from non-solving-oriented groups pass, but one or more other tests fail: 2/3
At least 50% of the tests from non-solving-oriented test groups pass: 1/3
Anything else: 0/3