Drill Problem #1

Function Name: replaceNaN

Inputs:

1. (*double*) An array with NaN and integer values
2. (*double*) An array of only integer values

Outputs:

1. (*double*) An array with integers replacing the NaN's

Function Description:

NaN is a value of class "double" that means "Not a Number." This value often occurs in MATLAB when undefined mathematical operations are performed, such as subtracting infinity from infinity or dividing infinity by infinity. NaN is often used to indicate something is of the class "double" but cannot be assigned a value.

In this problem, the first input array contains unwanted NaNs. You must write a function that replaces the NaN values in the first input array with the values in the corresponding indices of the second input array.

Example:

```
arr1 => [1, NaN;
          3,   4]
arr2 => [5,   2;
          7,   6]

result = replaceNaN(arr1, arr2)
result => [1, 2;
           3, 4]
```

Notes:

- The two input arrays are guaranteed to have the same dimensions.

Hints:

- You may find the `isnan()` function useful.

Drill Problem #2

Function Name: tempCheck

Inputs:

1. (*double*) An MxN array of temperatures recorded at the end of month 1
2. (*double*) An MxN array of temperatures recorded at the end of month 2
3. (*double*) An MxN array of temperatures recorded at the end of month 3

Outputs:

1. (*double*) An MxN array of average values over the quarter
2. (*logical*) An MxN logical array giving the locations where the thermometers are broken

Function Description:

We here at Global Dynamics have undertaken an experiment over the last quarter that tests the effects of workplace temperature on an individual's overall productivity. While you and your coworkers may have been confusedly uncomfortable, we feel that, ethics aside, the results were better for not having told any of the participants in the trial of what was happening. Now that we have the data, we need you to take the final temperatures recorded at the end of each month in the quarter and find the average temperature overall. We know that the temperature in each office changed at least slightly over the course of the trials, so any Super-Stealth-Undetecto-Thermometers that read the same temperature across the board were broken, and we need you to identify them in your final analysis.

You will be given three MxN arrays that will contain all of the thermometer readings at the end of each month. Your job is to average all three values for each position in the arrays. Any values that remain the same in a given position across all three of the arrays should not have an average displayed, since that means the thermometer is broken; instead the final average array should have a zero at each of those indices. You should also output a second array, this one of class `logical`, which has a `true` at each index where the broken thermometers are.

Notes:

- If your second output is not class `logical` you will not receive points for this problem.
- The input arrays will all be the same, unknown (MxN) size.
- Do not worry if the averages are nowhere near any of the values contained within the arrays: we here at Global Dynamics pride ourselves on the sheer variety of our test parameters.

Drill Problem #3

Function Name: multiTable

Inputs:

1. (*double*) A number, N.

Outputs:

1. (*double*) An NxN array that represents the corresponding multiplication table

Function Description:

Remember in 3rd grade when you had to memorize your multiplication tables? Remember how you thought you'd never have to do that again, only to have 4th grade prove you wrong? Well, soon you really never will have to memorize them, again, because you'll always have MATLAB to generate a multiplication table for you!

Write a MATLAB function that takes in a number, N, and returns an array of a multiplication table that goes from 1 to the number N. To illustrate, if you were given the number 5, then the multiplication table would look like the following (only as an array in MATLAB):

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

Notice that the array starts with 1 in the top left corner, counts up to N in the top right and bottom left corner, and has a value of N^2 in the bottom right corner.

Hints:

- You may find transposing useful. Remember, in order to transpose `arr1`, use a single apostrophe such that `arr1Transposed = arr1'`.

Drill Problem #4

Function Name: runBabyRun

Inputs:

1. (*double*) Target distance to run before half-marathon
2. (*double*) A 4x7 array of miles run each day in the month of February

Outputs:

1. (*double*) The most improvement made between consecutive days
2. (*logical*) Whether you met your target distance for a single run
3. (*double*) 1x4 vector of how many total miles you ran each week

Function Description:

With this New Year, you decide that you are actually going to run that half-marathon that has been on your calendar for three years now. You have a month to prepare for your race, so you decided to record how many miles you run each day and aim to run a target distance before the race; if you can run this many miles, then you will feel officially half-marathon ready!

The month is now over and your race is tomorrow, so you decided to look back and see the progress that you have made.

First, check the greatest mileage increase made between any two consecutive days. Next, check whether the number of miles run on any one day met the target distance you set before training (the number of miles will “meet” the target distance if it is greater than or equal to the value). Finally, go back through your calendar and track how many miles you ran each week. Because you need to get a good night’s rest, you decided to write a MATLAB function to determine all of these things for you!

Notes:

- The month during which you are training will always have 28 days in it (it is not a leap year), with each row containing the data for an entire week.

Drill Problem #5

Function Name: sudoku

Inputs:

1. (*double*) A 9x9 array of a potentially solved sudoku board

Outputs:

1. (*logical*) A logical output establishing whether the board is legitimately solved

Function Description:

Sudoku is a popular puzzle where the solver is given a 9x9 tile board with nine 3x3 tile sub-sections within the board. For an eloquent description of Sudoku, visit http://en.wikipedia.org/wiki/Sudoku.

Write a function that will input a completely filled Sudoku board and will output a logical value verifying whether the board is adequately solved according to Sudoku rules. This means each row, column, and 3x3 subsection needs to be checked that it contains the numbers 1 through 9 independently.

An example, correctly solved Sudoku board for reference:

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

Notes:

- You may find the `all()` and `sort()` functions useful.