CS1133, Spr 2018, HW 9 (Lect 21, Pts=100)

Due April 17, 2018 (before 11:00 pm)

Problem 1 [100 pts]

In this problem, we want to process a given matrix of UPC codes. You can reuse our solutions for previous assignments involving UPC codes, but this time you must separate out various parts of the program into a number of user-defined functions.

Your script program must be named `UPCcorrection`. It takes a given matrix of UPC codes, one per row, calls the function `ProcessAllUPC_` to process all the codes, and displays output results on the command window. Here is an example of the resulting displays:

```
Original UPCs:
     7     2     2     8     3     5     7     9     0     1     4     6
   NaN     2     2     8     3     5     7     9     0     1     4     6
     7     2     2     8     3     5     7     9     0     1     4   NaN
     7     2     2     8     3   NaN   NaN     9   NaN     1     4     6
     7     2     2     8     3     5   NaN     9     0     1     4   NaN
     2     8     3     5     7     9     0     1     4     6   NaN     8
     0     1     4     6   NaN     8     5     9     3     3     3     2
   NaN     8     5     9     3     3     3     2     9     0     1     2
     1   NaN   NaN     9     3     3     3     2     9   NaN     1     2
     9     2     8     3     5     7     0     1     4     6   NaN     1

Resulting UPCs:
     7     2     2     8     3     5     7     9     0     1     4     6
     7     2     2     8     3     5     7     9     0     1     4     6
     7     2     2     8     3     5     7     9     0     1     4     6
     7     2     2     8     3   NaN   NaN     9   NaN     1     4     6
     7     2     2     8     3     5   NaN     9     0     1     4   NaN
     2     8     3     5     7     9     0     1     4     6     5     8
     0     1     4     6     2     8     5     9     3     3     3     2
     1     8     5     9     3     3     3     2     9     0     1     2
     1   NaN   NaN     9     3     3     3     2     9   NaN     1     2
     9     2     8     3     5     7     0     1     4     6     4     1

Total number with 0 NaN is: 1
Total number with more than 1 NaN is: 3
Total number with 1 NaN is: 6
They are located in row(s): 2   3   6   7   8  10
```

Note that none of the functions defined here should produce any direct output displays. You can simply create the given matrix of UPC codes in an assignment statement.

The function `ProcessAllUPC_` must called the function `Process1UPC_` repeatedly, and must collect and produce results that the script program needs.

The function `Process1UPC_` processes just one UPC code and produces output results for the function `ProcessAllUPC_`. It uses two separate functions `FindNaNandSum1UPC_` and `Fix1UPC_` in order to perform its job.

The function `FindNaNandSum1UPC_` finds if a given UPC code has 0, 1, or more `NaN`s. If there is only one `NaN`, it records it position. The position is set as an `NaN` if there is 0 or more than 1 `NaN`. Also if there is only 1 `NaN`, then the function computes the weight sum of the remaining digits of the UPC code. The weighted sum has no particular meaning if the code has 0 or more than 1 `NaN`.

The function `Fix1UPC_` computes the missing value in the case of just 1 `NaN`, otherwise it does nothing. This function uses three separate functions `AtEvenPosition_`, `missingDigitAtOddPosition_` and `missingDigitAtEvenPosition_` to do its job.

The function `AtEvenPosition_` determines if the position of the `NaN` is located at an even position, and if so, it returns a value of true (otherwise it returns false).

The function `missingDigitAtOddPosition_` computes the missing digits at an odd position, and `missingDigitAtEvenPosition_` computes the missing digits at an even position.

Make sure that your functions have the necessary input arguments that it needs to accomplish its job. Your functions must also have the necessary output arguments so that the script program or other functions that call it get the results that they need.

Remember to zip your script program together with all the functions that you created into a single file. Then submit the resulting file.