CS1133, Spr 2018, HW 7 (Lect 14, Pts=40+40)

Due March 27, 2018 (before 11:00 pm)

Problems in part 2 of the course must be done using only scalar operations and functions. Vectors and matrices can be created, but you can only operate and manipulate one element at a time. Please read the lecture notes on Vectorization versus Scalarization.

Problem 1 [40 pts]

Given a single UPC where one and only one of the digit is missing or undetermined, write a program to compute its value. The digits are given as elements of a row vector, and the missing/undetermined digit is represented by `NaN`. That `NaN` can be located at an even or odd position. Produce the following output displays:

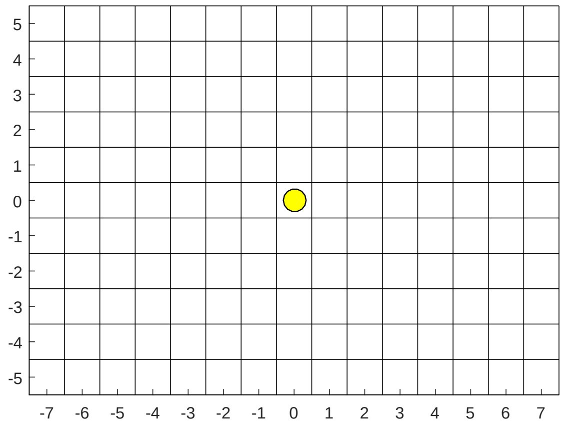
```
The NaN is located at position 3
Its value is computed to be 2
The corrected UPC is:  7  2  2  8  3  5  7  9  0  1  4  6

The NaN is located at position 4
Its value is computed to be 8
The corrected UPC is:  7  2  2  8  3  5  7  9  0  1  4  6
```

Problem 2 [40 pts]

A particle moves randomly inside a rectangular box consisting of a grid of squares, each 1 by 1 in size. The square at the lower left-hand side of the box is centered at the coordinates (x_1, y_1). The square at the upper right-hand corner of the box has coordinates (x_2, y_2).

At each time step, the particle has probabilities p_E, p_N, p_W, and p_S respectively for moving east, north, west, and south. The speed takes on a random but discrete value $1, 2, \ldots, 6$, with equal probability.

Special treatment is needed when an intended move would take the particle outside the box. In this problem, we assume the bouncing boundary condition, which forces the particle to bounce backward when a boundary is reached. One can consider the boundaries of the box to be at $x = x_1 - 0.5$, $x = x_2 + 0.5$, $y = y_1 - 0.5$, and $y = y_2 + 0.5$. Here we assume that $x_1 = -7$, $x_2 = 7$, $y_1 = -5$, and $y_2 = 5$ (as shown in the figure here).



For example, if the particle is currently located at the square centered at $x = 5$, and the following move is to the east at a speed of 6 (6 square to the right), it would end up at the square centered at $x = 11$, which lies outside the box. Because of the bouncing boundary condition, the particle turns around at the boundary and moves back the other way. Thus the particle moves from 5 to 6 and 7, turns around upon hitting the boundary and gets back to 7, and then to 6, 5, and arrives at 4. In general, if the current position is x, and the speed to the right is s, and if $x + s$ lies the right of the boundary at $x_2 + 0.5$, then the particle bounces back and ends up at $2x_2 - (x + s - 1)$. The extra term of -1 has to do with the discreteness of the grid structure.

Next, we consider a move where the particle would hit the boundary on the left at $x_1 - 0.5$. For example, if the particle is currently at $x = -5$, and the move happens to be to the left (west) at a speed of $s = 6$, then it would end up at $x - s = -11$, which lies outside the left boundary. The bouncing back condition means that the particle moves from -5, to -6, and -7, hits the boundary and turns back to -7, and then proceeds to -6, -5, and finally ends up at -4. In general, if the current position is x, and the speed to the left is s, and if $x - s$ lies the right of the boundary at $x_1 - 0.5$, then the particle bounces back and ends up at $2x_1 - (x + s + 1)$. Again the extra term of 1 has to do with the discreteness of the grid structure.

With straightforward modification, the above results apply to movement along the y-axis as well. That should take care of the particle bouncing off the top and bottom boundaries of the box.

In general, we assume that x_1, x_2, y_1, and y_2 have large enough magnitudes compared with the maximum speed, so that there is at most one bounce from a boundary in each move (no multiple bounces).

Write a program (using a for-loop) to run a simulation involving a total of 11 time steps (moves) with the particle starts at the central square at $(0, 0)$ (as shown by the yellow circle in the figure). Assume that $p_E = 0.41$, $p_N = 0.22$, $p_W = 0.15$, and $p_S = 0.22$. Use a vector to stores these 4 probabilities. You need to use the function `cumsum` (cumulative sum) to cumulatively sum up its elements. Use the help in Matlab to find out what this function does. Please read page 17 in the lecture notes on Probability, Statistics, and Random Numbers for ideas how to simulate the choice of the direction of movements.

Here is the output displays of one such simulation:

```
 timeStep   rand#     speed   xPosition  yPosition
===================================================
   1.0000    0.1499    5.0000    5.0000         0

   2.0000    0.5186    4.0000    5.0000    4.0000

   3.0000    0.6490    6.0000   -1.0000    4.0000

   4.0000    0.4538    5.0000   -1.0000    2.0000

   5.0000    0.8253    3.0000   -1.0000   -1.0000

   6.0000    0.1332    1.0000         0   -1.0000

   7.0000    0.3909    2.0000    2.0000   -1.0000

   8.0000    0.8034    5.0000    2.0000   -5.0000

   9.0000    0.3993    1.0000    3.0000   -5.0000

  10.0000    0.4168    4.0000    3.0000   -1.0000

  11.0000    0.6280    4.0000    3.0000    3.0000
```

The random number as shown above was used to choose a direction of movement. `xPostion` and `yPosition` give the location of the particle at the end of that time step. Your program needs to produce similar output displays.