1. Let’s define a function R(n) such that it is a function that outputs a random integer in the range 1 to n. Your task is, given the function R(5) and using basic algebra (addition, subtraction, multiplication), define the function R(7). Explain your algorithm. Note: Likelihood of every output (1, 2, 3, 4, 5, 6, 7) has to be the same (1/7).

2. You have been given a maze in the form of the matrix that has M columns and N rows. Matrix consists of the 1s and 0s, 1 being a field that you can step on, and 0 being a field that you are not allowed to step on. Legal moves are right, down, left and up to the fields next to the field that you are on. You start at the top left corner, and the exit is at the bottom right corner. The example of the maze is given below:

   1 1 1 1 0
   0 1 0 1 0
   1 1 0 1 0
   1 0 0 1 0
   1 1 0 1 0
   1 1 1 1 1

   The algorithm below finds the shortest path through the labyrinth. Your task is to track
the
   algorithm for the example given above.

   The algorithm uses one queue (let’s call it Q), and a one dimensional array of M times N (dimensions of the labyrinth) elements. First element in the array is associated with the entrance in the labyrinth, and the last element is associated with the exit, and every element in between (all 1s and 0s) has an associated index in the array. For the example it would be:

   0 1 2 3 4
   5 6 7 … 29

   At the beginning, all elements of the array are set to -1 (meaning that you did not step on that position in the labyrinth). So you start at the entrance (top left corner), and you check the neighboring elements(for the sake of this example, let’s set the priority to be right-down-left-up). If an element is 1, and it was not approached before (you check the array for -1), you put the appropriate index of the element in Q, and on its position in the array you write the index of the element that you are currently on. In the example given above, that would mean that you start at the entrance, see that you can go right and array[1]==-1. Therefore, you put 1 in Q, and set array[1]=0.
   After you checked all neighboring elements, you dequeue Q, and get the element X. If X is the exit, you are finished, and you track backwards elements of the array to find the path Otherwise, you check neighboring elements of X.

Note: If X is the exit, you check the last element of the array, and let’s say its value is Y. Then you go to array[Y], get its element, and follow the recursion until you reach the array[0]. All indexes that you checked are the shortest path out of the labyrinth.

For today’s exercise, you need to track the algorithm for the example above, and show the final state of the array and Q. Also, explain why the algorithm finds the shortest path.