5. **Apply BFS/DFS/MST to solve a problem (Portfolio Project Problem):**
   You are given a 2-D puzzle of size MxN, that has N rows and M column (N>=3 ; M >= 3; M and N can be different). Each cell in the puzzle is either empty or has a barrier. An empty cell is marked by '-' (hyphen) and the one with a barrier is marked by '#'. You are given two coordinates from the puzzle (a,b) and (x,y). You are currently located at (a,b) and want to reach (x,y). You can move only in the following directions.
   L: move to left cell from the current cell
   R: move to right cell from the current cell
   U: move to upper cell from the current cell
   D: move to the lower cell from the current cell
   You can move to only an empty cell and cannot move to a cell with a barrier in it. Your goal is to find the minimum number of cells that you have to cover to reach the destination cell (do not count the starting cell and the destination cell). The coordinates (1,1) represent the first cell; (1,2) represents the second cell in the first row. If there is not possible path from source to destination return None.
   Sample Input Puzzle Board: [[-,-,-,-,-],[-,-,#,-,-],[-,-,-,-,-],[#,-,#,#,-],[-#,-,-,-]]

-	-	-	-	-
-	-	#	-	-
-	-	-	-	-
#	-	#	#	-
-	#	-	-	-

   Example 1: (a,b) : (1,3) ; (x,y): (3,3)
   Output: 3
   On possible direction to travel: LDDR
   (1,3) →(1,2) → (2,2) → (3,2)→(3,3)

   Example 2: (a,b): (1,1) ; (x,y): (5,5)
   Output: 7
   One possible direction to travel: DDRRRRDD
   (1,1) →(2,1)→(3,1)→(3,2)→(3,3)→(3,4)→(3,5)→(4,5)→(5,5)

   Example 3: (a,b): (1,1); (x,y) : (5,1)
   Output: None

a. Describe an algorithm to solve the above problem.
b. Implement your solution in a function **solve_puzzle(Board, Source, Destination).** Name your file Puzzle.py
c. What is the time complexity of your solution?
d. **(Extra Credit):** For the above puzzle in addition to the output return a set of possible directions as well in the form of a string.