Problem 1:

You are given 3 objects as inputs to a game: a Grid (of arbitrary size), an integer K, and an array of strings called “Rules”.

1. The Grid is an array of arbitrary size consisting of “0” and “1”, e.g., ([[1, 0, 1, 0], [0, 1, 0, 1]] .
2. K: an integer denoting the number of rounds you will need to play for this game. For instance, K =5;
3. Rules: an array of arbitrarily ordered strings of ‘dead’s and ‘alive’s. For instance, Rules = ['dead', 'alive', 'dead', 'alive', 'dead', 'dead', 'dead', 'dead', 'dead' , 'dead', 'dead', ……].

Game procedure is explained as follows:

1. Let the initial Grid be given by the following 2x4 array. K = 2.

Grid (initial grid):

1	0	1	0
0	1	0	1

2. Let Rules = ['dead', 'alive', 'dead', 'alive', 'dead', 'dead', 'dead', 'dead', 'dead' , 'dead', 'dead', ……]. Represent Rules by a Python dictionary Rules_Indication:
    a. Rules_Indication: key is the index of Rules, value to the key is based on ‘dead’ = 0 or ‘alive’ = 1
    b. Rules_Indication = {0: 0, 1: 1, 2: 0, 3: 1, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0, …… }

3. Find the sum of nearest neighbors for each cell in the initial grid, shown in Grid_Neighbors

Grid_Neighbors:

1	3	2	2
2	2	3	1

4. Transform the Grid_Neighbors based on Rules into Grid Transformed. If you have number 3 in Grid_Neighbors, based on rule, the transformed number should be 1 where you can find from Rules_indication.

Grid_Transformed (new initial grid for next round play)

1	1	0	0
0	0	1	1

5. The Grid_Transformed now will become the new initial grid for the next round play.

Task: Write a Python script gridGame to generate the outcome of playing the game for K rounds. The output of function gridGame should be the Grid_Transformed from last round after the Kth round.

In the above example, the final grid after K = 2 rounds would be:

1	0	1	0
0	1	0	1

```
def gridGame(Grid, K, Rules):
        … coding …
        return final_grid
```

Problem 2:

An array is given with arbitrary shape (for example, a 5 X 3 array as given below) as input for function findIndex, in the meantime, rank (an integer) is also given as another input for function findIndex. The prototype of the function is as follows. Please complete this function and provide 2 test cases to valid the function output.

```
def findIndex(Array, Rank):
        … coding …
        return (rank index)
```

output: 3

For input value Rank = n, you first need to find the row-sum of the given Array and store it in another array, say, with name Sum_Array. The function findindex shall return the index of the n-th largest value in Sum_Array. Namely, the returned value of function findIndex shall be an integer which corresponds to the index of the element with Rank=n in Sum_Array.

In the example below, if the input array is Array and the rank value is 3, then you need to find the index of the third largest number in the Sum_Array. Your code shall return number 4 (fifth element in Sum_Array). In this case, 88 appears twice in the Sum_Array, and the first 88 in the Sum_Array ranks higher than the second 88 (2 is higher than 3 in the rank definition). Thus the index of the second 88 is returned.

```
Array = [[21,31,23],
        [32,14,42],
        [3, 5, 7],
        [25,37,27],
        [29,17,42],]

Rank = 3

Sum_Array =  [[75],
              [88],
              [15],
              [89],
              [88]]
```