1. Give the matrix size *n* – square matrix *n*x*n*
2. Grid of a matrix to input the data
3. Give the row no that will be perturbed/changed - e.g. row 2, row 3
4. Possible values of the perturbations – increments of 0.1 between -1 and 1 (see Excel file for values). For incremental grid of 0.1 there are 21 x 21 = 441 possible cases. So I need to generate all the possible cases as follows: if row 2 is perturbed by 0, then row 3 can be perturbed by a value between -0.95 to 0.95. (see Excel file for example of possible combinations)
5. If perturb for row 2 is *x*, now we need to change all the values on the row using the following rule:
   * If , then the new value on the matrix entry is , otherwise (see Excel file)
   * The perturbation *x* will be apply only for the cells in row 2 >= 0.
6. We normalize the perturbed matrix obtained in 5 to maintain the rule of sum over columns = 1. We will do it as follows: . (see Excel file)
7. The normalized matrix from point 6 is raised to power 64 (see Excel file).

Matrix ( int rowsAndColumnsValue, int power)

{

for(int a=0; a< rowsAndColumnsValue; a++){

for (int b=0; b< rowsAndColumnsValue; b++){

pow(matrix[a][b],power);

}

}

1. Using the matrix A^64 we calculate the priorities which will be based on the values of the last column of the matrix from row 4 to the last one – the rest of the rows (the area is highlighted in blue in the Excel file). The perturbations are calculated as follows: . The sum of the priorities obtained has to be equal to 1. (see Excel file)
2. As final output, I need to be printed on the screen the values of the perturbations and the values of the priorities – e.g. perturbation 1 = 0.5; perturbation 2 = -0.3; priority 1 = 0.37278; priority 2 = 0.39121; priority 3 = 0.23601.
3. I need all 441 possible combinations and I also need to be able to export the values obtained at 9 to an Excel file after computations.