Homework 2

Submission Format:

Submit a zip named <Roll_Number>.zip which on unzipping should have the following directory structure.

```
----<Roll_Number>
            |----------q1
                      |---- <Roll_Number>_sc.py or <Roll_Number>_sc.cpp.
                      |---- README.md
                      |---- <Roll_Number>_report.pdf
```

Q1 -> Programming question: **your program must assume that the large dataset does not fit in main memory**, the report should contain your plots and observations, readme should contain instructions to run your program and file format used (bin/txt).

Execution and Input/Output format:

```
python <Roll_Number>_sc.py <matrix_file_1> <matrix_file_2>
./a.out <matrix_file_1> <matrix_file_2>
```

(Text File Format)

Input format:

N, number of rows: 1<=N<1e5
N will take up 5 positions (with leading zeroes, eg 1 will be represented as “00001”, ie, **fixed width of 5**)

mat[i][j], matrix elements: 0<=mat[i][j]<=9
Matrix elements will take up 1 position (**fixed width of 1**)

File format:

content: N followed by N lines (total N+1 lines):
mat[i][0]<space>mat[i][1]<space>...mat[i][N-1]<newline>

N
mat[0][0] mat[0][1] … mat[0][n-1]
mat[1][0] mat[1][1] … mat[1][n-1]
…
mat[n-1][0] mat[n-1][1] … mat[n-1][n-1]

The first 5 positions represent the value of N, skipping the next position (‘ ’), the next 1 position represent mat[0][0], skipping the next position (‘ ’), next 1 position represent mat[0][1], and so on… the next 1 position represent mat[0][n-1], skipping the next position (‘\n’) and so on.

Output format:

ans[i][j], matrix elements: 0<=ans[i][j]<1e7
Matrix elements will take up 7 positions (with leading zeroes, eg. 1 will be represented as “0000001”, ie, **fixed width of 7**)

naming: <Roll_Number>_out.txt
file format:
N lines: ans[i][0]<space>ans[i][1]<space>...ans[i][N-1]<newline>

ans[0][0] ans[0][1] … ans[0][n-1]
ans[1][0] ans[1][1] … ans[1][n-1]
…
ans[n-1][0] ans[n-1][1] … ans[n-1][n-1]

The first 7 positions represent ans[0][0], skipping the next position (‘ ’), next 7 positions represent ans[0][1], and so on… the next 7 positions represent ans[0][n-1], skipping the next position (‘\n’) and so on.

(Binary File Format)

Input format:

N, number of rows: 1<=N<1e5
N will take up **4 bytes**.

mat[i][j], matrix elements: 0<=mat[i][j]<=9
Matrix elements will take up **1 byte**.

File format:
matrix represented in row major format.

content: N followed by (N*N numbers: <row1><row2>...<rown>).

<row 0> = <mat[0][0]><mat[0][1]> … <mat[0][n-1]>
<row 1> = <mat[1][0]><mat[1][1]> … <mat[1][n-1]>
…
<row n-1> = <mat[n-1][0]><mat[n-1][1]> … <mat[n-1][n-1]>

<N><row 0><row 1>...<row n-1>

The first 4 bytes represent the value of N, the next 1 byte represent mat[0][0], next 1 byte represent mat[0][1], and so on… the next 1 byte represent mat[0][n-1] and so on.

Output format:

ans[i][j], matrix elements: 0<=ans[i][j]<1e7
Matrix elements will take up **4 bytes**.

naming: <Roll_Number>_out.bin
file format: N*N numbers in row major format:

<row 0> = <ans[0][0]><ans[0][1]> … <ans[0][n-1]>
<row 1> = <ans[1][0]><ans[1][1]> … <ans[1][n-1]>
…
<row n-1> = <ans[n-1][0]><ans[n-1][1]> … <ans[n-1][n-1]>

<row 0><row 1>...<row n-1>

The first 4 bytes represent ans[0][0], next 4 bytes represent ans[0][1], … the next 4 bytes represent ans[0][n-1], and so on.