Distributed Systems
Monsoon 2017 Assignment-4

1) Data Marshalling : To write programs and perform comparison between Json and Google Protobuf for serialization and deserialization of given set of records. Comparison is to be done with respect to time and rates of (de)serialization

Input Record Format(input.txt):
Example:

xyz,202821785:cs210,85:cs155,94
pqr,202817265:cs210,20:cs155,47

Write a RMI client-server program to transfer the input.txt file from client to server. The input.txt should be (de)serialized using the below formats. Write the records transferred in the output.txt file on server side.

Json format: Read the input.txt file and convert the records . An illustration for the serialized version of the input records above is as shown:

[{ "Name": "xyz", "CourseMarks": [{ "CourseScore": 85, "CourseName": "cs125" }, { "CourseScore": 94, "CourseName": "cs210" }], "RollNo": 202821785 }, { "Name": "pqr", "CourseMarks": [{ "CourseScore": 20, "CourseName": "cs125" }, { "CourseScore": 37, "CourseName": "cs210" }], "RollNo": 202817265 }]

Google protobuf format :
Convert the records to the given Google protobuf format(Result.proto).

General Instructions :-

For both the formats print the time taken and rate of Serialization/Deserialization on client/server side. Rate of (de)serialization is defined as :
Amount of Data Converted/Time taken for conversion.

The RMI function on the server side would be of the form Receive(byte[]) where byte[] is the serialized form of the records.

Time should be measured in milliseconds (ms). The Time taken for conversion should not include the Time taken for File I/O operations. Result.proto, a sample input file and its corresponding result.json is attached for reference. The input.txt and output.txt files should have the **exact same content**.

Follow the instructions and tutorials given here for installing and understanding google protobuf: https://developers.google.com/protocol-buffers/

2) Game of life :-

Implement the Game of life using MPI and the rules provided. Each cell of the matrix is in one of 2 states ie. dead or alive.
See https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life for more information on game of life.

```
mpiexec -np N prog seed.txt T
```

Here seed.txt is the input file and T is the number of timesteps. Max possible size of matrix is **24x24** .

Input :-

A pattern is provided initially(seed) as input consisting of 0's(dead) and 1's(alive). This has to be read from a file.

Example :-

```
01100
01001
11010
11001
00010
```

Rules of Game of life :-

1. Any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
2. Any live cell with two or three live neighbours lives on to the next generation.
3. Any live cell with more than three live neighbours dies, as if by overpopulation.
4. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

Output :-
Print the current timestep along with the game of life matrix using 0's and 1's for T steps.

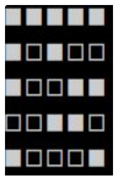
Example :-

```
t1
01100
```

```
00010
00011
11011
00000
.
.
.
tT
00000
01001
01000
10011
00100
```

Bonus :-

Print the game of life matrix use opaque blocks as shown



Submission Format :-

1. Roll No. (Folder)
    a. Q1 (Folder)
        i. <All code related to Q1>
    b. Q2 (Folder)
        i. <All code related to Q2>

Submit archive as "Assignment4.tar.gz".

Ps: Plagiarism will be heavily penalised.