Stochastic Systems

For this assessment I ask you to implement Markov chains and to answer 6 questions relating to those. In order to successfully complete this assessment you will have to write a Java code to formulate Markov chains.

For the assessment, I give you two files. `Markov.java` and `Stochastic.java`. The latter class contains the main method, in fact it only contains the main method from which it calls a number of methods in `Markov.java.` It also contains comments which formulate the questions you need to answer in the form of computer code. You need to read `Stochastic.java` in detail to understand how to implement the relevant methods. It also contains the questions that you need to answer.
Altogether, there are 6 questions you need to answer by writing code. I reproduce only the first one here:

```
//Q1
//Assume a square 3x3 grid on which a turtle performs a random walk in discrete time.
//The turtle can walk east, west, south, north.
//Translate the grid into a Markov chain.
//Assume that the desired steady state probabilities of all states are equal.
// Return  as A1 the  transition probability that the turtle walks from state s1 to state s2
//You must use the method we discussed in the lectures. I do not accept other answers
(even if correct).

//numStates .... number of states of the Markov state.
//s1, s2 ...  the states of the Markov chain.
int  numStates =9;
int s1 =1;
int s2 = 2;

double A1 = Markov.getTransProb(s1,s2,numStates);
```

Here, you are asked to implement the method getTransProb which should return as a single number the transition probability from state s1 to state s2. Note the directionality here. At the moment the parameters are set such that A1 would be the transition probability from state 1 to state 2 for a Markov chain of size 9. <u>Yet, during marking these numbers will change. You must take this into account when coding getTransProb.</u>

You can edit `Stochastic.java` but bear in mind that all changes you make to it will be lost. During marking, I will replace it with a different version. I therefore recommend that you only edit the numerical values of the parameters, not the rest of the code. **Your solution should be programmed in Markov.java only.** In fact, you do not even need to submit Stochastic.java. You can also add other class files for as long as they are called from within `Markov.java` and the program compiles as specified below. The version of `Markov.java` I give you is only the shell of a class which compiles, but actually does not do anything useful. Your task is to change that.

Submission

- When marking, I will use the same problem to assess your code, but the numbers will be different from the examples that are currently in the code (unless indicated otherwise).

- Most importantly, I will replace `Stochastic.java` with a different version. Any code you place there will be lost. You can make changes to the file, but any code you add there will be disregarded.
- Submission is to raptor into the folder `/proj/comp[modcode]/stochastic/xyz.` Replace [modcode] by the module code (i.e. 8370 or 6370 as appropriate) and xyz by your login.
- Your java program must compile with the original version that you download from moodle. More spcifially, it must compile on raptor from the command line like so:

  ```
  javac -cp . *.java
  ```

  It must then be possible to execute the program from the raptor command line like so:

  ```
  java Stochastic
  ```

  Before submitting, make sure that this works on raptor. If I cannot compile or start your program using these above commands, then you **will** get 0 marks.
- Please ensure that your program is located in the top level of your folder, i.e. in `/proj/comp[modcode]/stochastic/xyz` and not in a subfolder. This means that compilation and execution must work when I am in the directory `/proj/comp[modcode]/stochastic/xyz`. Failure to follow this instruction will incur a penalty in marks.

Your program must complete within 100 seconds. If it does not, it will be killed and you get 0 marks. Note, that the duration of programs is stochastic. So you should make sure that the version you submit runs within much less than 100 seconds (say 80) to be on the safe side.

Marking criteria

Your final submission will be assessed by compiling and running it on raptor using the above two commands. It is your responsibility to ensure before submission that they work and produce the desired result. **If this does not work, you will be awarded 0 points.**

Non-compiling/non-running programs will get 0 marks.

Rules on use of APIs: Stochastic`.java` includes, at the moment

```
java.lang.Math
```

and

```
java.util.*
```

These allow you to use mathematical functions in java and ArrayLists. You are not allowed to use any APIs beyond that. In particular, do not use any graphics or GUI. This is not necessary and not allowed.

- You will get up to 40 mark for a program that compiles, runs, displays functionality and completes within 100 seconds on raptor.
- You will get up to 10 mark for each question that is answered correctly.

Deadline

The deadline for submission see Moodle. The submission folder closes at 23:40 on the day of the deadline. Do not leave it to the last second to submit, but aim to submit at least a day before the deadline to iron out technical problems, network issues, etc...

Section 2.2.1.1 of Annex 9 of the Credit Framework states that, "Academic staff may not accept coursework submitted after the applicable deadline except in concessionary circumstances".

A Frequently Asked Questions document on Plagiarism and Collaboration is available at: `www.cs.kent.ac.uk/teaching/student/assessment/plagiarism.local`. The work you submit must be your own, except where its original author is clearly referenced. Checks will be run on submitted work in an effort to identify possible plagiarism, and take disciplinary action against anyone found to have committed plagiarism. When you use other peoples' material, you must clearly indicate the source of the material.