Homework 4 – Banker's algorithm

Implement the Banker's algorithm for deadlock avoidance, with a given set of N processes ($N<10$, processes are P_1, P_2, …, P_N) and M resource types ($M<10$, resources are R_1, R_2, …, R_M). Use Java or C/C++ for the implementation, with a simple interface, where the user only supplies the name of the input file (text file, say "input.txt"). The program reads all the necessary input data from that file. You are free to choose the format of the input file, just make sure it contains all the necessary data. The input data and the result of the algorithm must be displayed on the screen.

The pseudo code for the Greedy version for the Banker's algorithm can be found in this module's Commentary. We know that this algorithm only finds ONE solution (safe sequence of processes) assuming there is one; otherwise reports there is no solution.

You must **adjust** this algorithm to find exactly TWO solutions instead of just one (assuming of course there are at least two solutions). If the algorithm only has one solution this must be reported as such and the single solution must be displayed.

To resume, there are only three possible scenarios, for a correct input file:

1. There is no solution (no safe sequence). The program must report there is no solution.
2. There is EXACTLY ONE solution (safe sequence). The program must report there is exactly one solution. The solution must be displayed on screen.
3. There are TWO OR MORE solutions (safe sequences). The program must find EXACTLY TWO solutions. The solutions must be displayed on screen. If there are more than two solutions, it doesn't matter which ones are found, as long as they are exactly two.

Note 1: the solution must be based on the Greedy version for the Banker's algorithm; in particular, using an approach for finding ALL solutions and then keeping and reporting just two of them is NOT allowed.

Note 2: The input file may be incorrect, for various reasons. This must be checked and reported by your program before any attempt of finding a solution.

HINTS:
- The input files should start with N and M, then the necessary matrices
- Work on your own computer or online using https://replit.com

DELIVERABLES:
1. The source code for your program, stored in a text file
2. One incorrect input file, "input0.txt" and one or more screenshots showing the results of your program for this incorrect input.
3. Three input files, "input1.txt" (N=4, M=3), "input2.txt" (N=4, M=3), "input3.txt" (N=4, M=3). Please choose the input data in such a way that the first input has no solution, the second input has exactly one solution and the third input has at least two solutions.
4. Three screenshots showing the final results of your program's execution for these three input files.
5. Store the screenshots in graphic files of your preferred format(".png" or ".gif" or ".pdf", etc.).