Laboratory Activity 4

Banker's Algorithm

(Safety and Resource-request Algorithms)

DESCRIPTION

The objective of this laboratory activity is to create a simulator that demonstrates the behavior of the banker's algorithm (safety and resource-request algorithms).

WEEK 8

1. Search the Internet for the different ways of creating a simulator program for the banker's algorithm (safety and resource-request algorithms).
2. Create a GUI for a similar simulator program that follows the required specifications indicated in Appendix A.
3. Create a new title for your banker's algorithm simulator program but the banker's algorithm (safety and resource-request algorithms) should also appear.
4. Zip the source file and submit the file to submit2jpyusiong@gmail.com.
5. The subject of the email should follow this format: **[CMSC 125-G] Laboratory Activity 4A_FamilyNameoftheLeader.**
6. The body of the email must contain the names of the group members.
7. After one hour, each group has ten minutes to show their mockup to the class.

WEEKS 10-11:

1. Create three problems with solutions for the safety and resource-request algorithms included in our learning resource. Save the problems with solutions with the filename **Laboratory_4B_Problems.docx.** Be sure that the new problems are not copied from our learning materials.
2. Start learning the jar to exe conversion, and search about the tools related to the jar to exe conversion. These tools include launch4j and JSmooth, among others.
3. Implement the approved design for your banker's algorithm simulator.
4. Include instructions on how to use your simulator software.
5. The simulator should allow the user to use the mouse when using the software.
6. Compile your source code using the command line to check if all dependency files are contained in your game directory.
7. Include a ReadMeFirst.txt file containing the instructions on how to compile and run your program from the command line.
8. Create a standalone game application using the jar to exe conversion program, such as Launch4j or JSmooth, among others.
9. Upload the source file (in zip) and the executable version to the Google Drive and create a link.
10. Submit the link to the files (including the document containing the three problems with solutions to submit2jpyusiong@gmail.com.
11. The subject of the email should follow this format: **[CMSC 125-G] Laboratory Activity 4B_FamilyNameoftheLeader.**

12. The body of the email must contain the names of the group members.
13. After one hour, each group has ten minutes to demonstrate their simulator program in class.

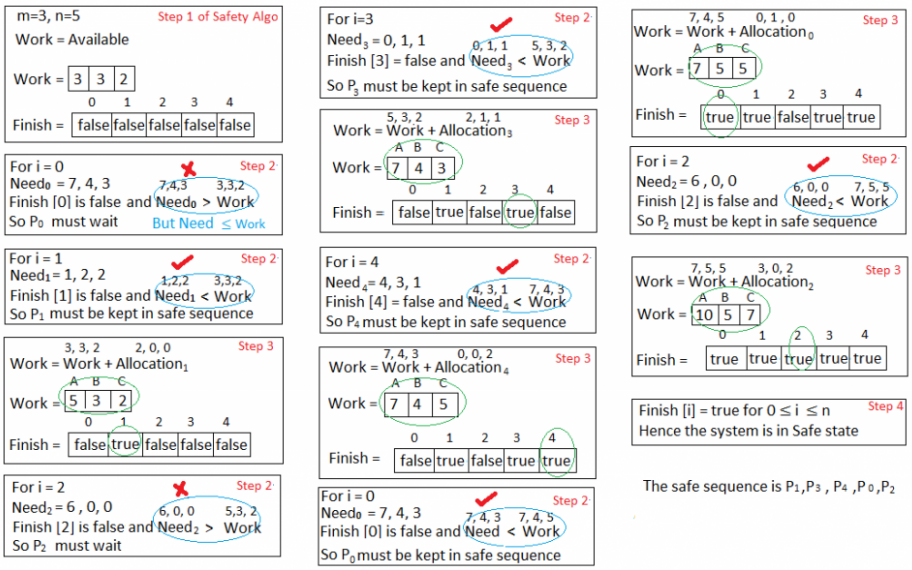
APPENDIX A (BANKER'S ALGORITHM)

1. A user has three choices on how to generate the data for the banker's algorithm: (random, user-defined input, user-defined input from a text file)
    a. **Random**: the program will randomly generate the data needed by the banker's algorithm.
    b. **User-defined input**: the user will input the data through an input screen in the simulator.
    c. **User-defined input from a text file**: the user will input the data by reading a text file.
2. The data needed by the banker's algorithm simulator are the number of processes, the number of types of resources present in the system, allocated resources to each process, the maximum claim, which indicates the maximum demand by a process, and the available resources. A sample window showing the data is presented below:

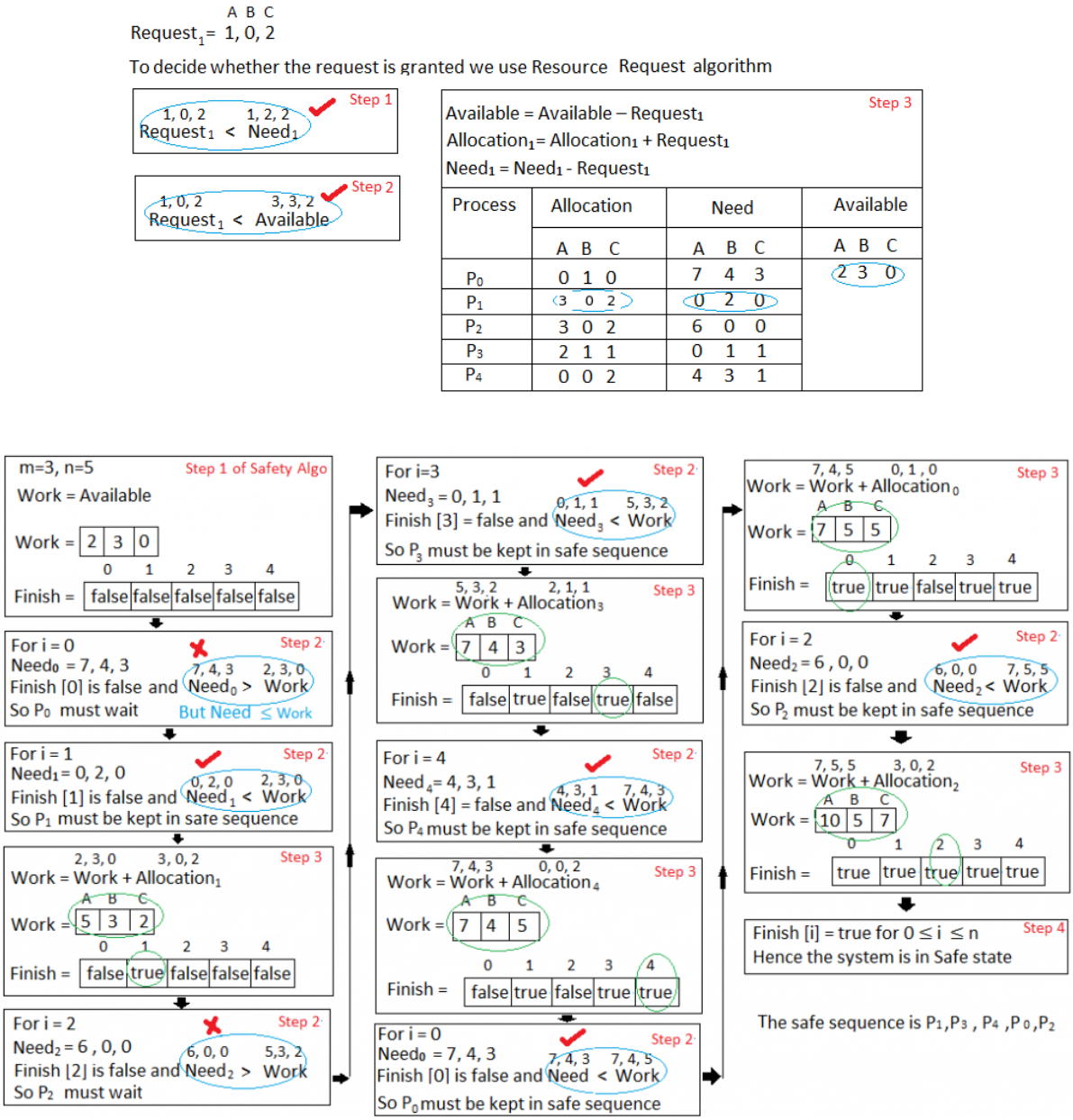
Process	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

3. For the resource-request algorithm, a request array containing the requested resources by a process is needed as input.
4. The safety algorithm simulation output has to show a table, but with the computed Need matrix and the step-by-step execution.

Process	Allocation	Maximum Claim	Need	Available
	A B C D	A B C D	A B C D	A B C D



5. For the resource-request algorithm simulation, the outputs are the following:



6. For additional details, please check this website: https://www.geeksforgeeks.org/bankers-algorithm-in-operating-system-2/