CPSC 351 Project 1 – Banker's Algorithm, due Saturday, 16 Mar 2019

Your name and company name: Michael Hounsworth CPSC 351 - 05

Repository https://github.com/ mhounsworth / Bankers_Project.git

Verify each of the following items with a corresponding checkmark. Incorrect items will incur a 5% penalty on the grade.

Complete	Incomplete	Banker's Algorithm
☒	☐	Created the Banker's Algorithm in C, C++, or Java, so that customers make requests of the banker for a limited set of resources.
☒	☐	Reads from and uses the current allocation state and maximum resources needed for each process (customer) from an input text file.
☐	☒	Has an **Interactive state**, that allows the user to type in a command of the form: * OR **<Rq | Rl> <process number> <R0 resources> <R1 resources> ...** **e.g., Rq 1 2 4 3** requests that process 1 be granted an allocation of 2, 4, 3 for R0,R1,R2, Rl ... releases them; * shows state of system (max, allocations, needed, available)
☒	☐	Has a **Simulation state**, that has processes make random requests for resources, and attempts to find a safe path for all processes to run to completion and release their resources. The simulation ends when all processes have shut down. (see examples)
☒	☐	Uses threads to simulate customers making requests and releasing resources when their maximum needs are satisfied
☒	☐	Uses Allocated, Maximum, Needs, and Available matrices to track the resources allocated, the maximum resources needed, resources needed and the resources available.
☒	☐	Does not grant requests that would exceed the maximum resources needed by a process.
☒	☐	Does not grant requests that would exceed the total resources available to the system.
☒	☐	Does not allow requests to be granted that would result in the system being in an unsafe state, by granting a proposed request and trying to find a path that satisfies all processes.
☒	☐	Tested the simulation with different input files that are known to have the system in a safe state to see if the simulation can find the solution (use Silberschatz and Stallings).
☒	☐	Tested the simulation with different numbers of resources (3-6), and different numbers of processes (3-10), and are able to describe the effect of doing this. Increasing resource types more denials to random resource requests Increasing processes system resources yet split, more processes waiting for resources
☒	☐	Tested the simulation to find the most likely combination of processes and resources to cause deadlock fewer or split resources & higher # of processes
☒	☐	Understand the code structure to the degree that the student could rewrite any section of the code from scratch.
☒	☐	Project directory pushed to new GitHub repository listed above using GitHub client.

Your comments

Interactive state exists, but not in the same form