CPSC 351 Project 1 – Banker's Algorithm, due Friday, 6 Nov 2020

Your name: Hai Nguyen

Repository *(print)***:** https://github.com/hai-nguyen93/cpsc35-bankers_algorithm

Verify each of the following items and place a checkmark in the correct column. Each item incorrectly marked will incur a 5% penalty on the grade for this assignment

Finished	Not finished	
X		Created the Banker's Algorithm in C, C++, or Java, so that customers make requests of the banker for a limited set of resources.
X		t 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.
	X	Reads from and uses multiple input text files to run the simulation. Each text file contains the resources for the Bank's Available resources on its first line, and the Allocated and Maximum resources for each Customer on its subsequent lines.
	X	Uses a Vector-like class, such as Vector<E> (Java), std::vector<T> (C++), or a subclass that allows vectors to be added, subtracted, and compared.
X		It uses a Bank class to simulate the Bank, which has a vector (Avail) that holds a set of resources that will be requested by Customers. The Bank determines if it is safe to allocate the resources to the customer, and approves the request or denies it.
X		It uses a Customer class, which has an Alloc vector to hold its resources, a Max vector to indicate the maximum number of resources it requires, and a Needs vector to indicate how many resources it needs to reach its maximum.
X		Uses threads to simulate customers making requests and releasing resources when their maximum needs are satisfied
X		Has Customers make random requests from the Bank, and has the Bank determine if the request is safe using the Banker's algorithm. Namely, it temporarily grants the request, then determines if there is still a path to satisfy all other Customers to reach their Max resources required. If there is, the request is approved; if not, it is denied.
X		Each time a request is approved, the Bank prints out the entire state of the system: its Avail vector, and each Customer's Alloc, Max, and Need vectors.
X		Once a Customer reaches its Max resources, it releases them all and shuts down its thread. The simulation runs until all Customers have shut down, or runs until it is in a deadlock it cannot recover from (e.g., the system's algorithm either fails, or the system starts in deadlock).
X		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).
X		Has a S**imulation 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)
X		Project directory pushed to new GitHub repository listed above using GitHub client.

Comments: - I only read 1 input file (infile.txt) per run because it is easier to track the output for me.
- I use array instead of a vector class or vector-like class.