# CPSC 351: OS - Spring 2019
## Project One, **Banker's Algorithm,** due Saturday, 16 Mar 2019

In this assignment, you will create a solution for the banker's algorithm project from section 8.6.3 (Banker's Algorithm) in the 10th edition of Silberschatz (Operating System Concepts). Your solution can be in either: C, C++, or Java. A partial, possible layout of your project in Java is laid out below. The description of this algorithm is in ch. 8 of Silberschatz, and in ch. 6 of Stallings.

Your simulation should contain the essentials of the banker's algorithm, namely, customers (threads) should make requests of the banker for a limited set of resources. The banker should check whether the customer is exceeding the maximum number of resources he/she said it would ever request, and if there are sufficient system resources, and if the request would put the system in an unsafe state.

To determine if the system would be in an unsafe state, the algorithm should attempt to grant the request, by increasing the resources allocated to the given process, and by reducing the total resources still needed by it to satisfy the maximum resources it needs. If there is a still a path of allocating resources and having them released that will satisfy all of the processes that are running, then the system is in a safe state, and the request should be approved.
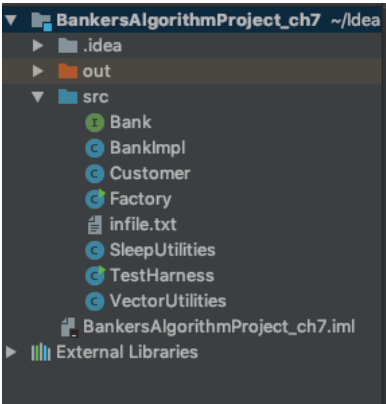
Only safe requests should be approved.

Once the request is approved, the state of the system (Allocated, Need matrices and Available vector) should be updated.

You can either have the customer make requests until they reach their maximum needed, then release all resources and exit, OR have the customers randomly request and release resources, sleeping after each request.

### Submission

Turn in the code for this project by uploading all of the source files you created to a single public repository on GitHub. While you may discuss this homework assignment with other students, work you submit must have been completed on your own.

To complete your submission, print the sheet at the back of this file, fill out its spaces, and submit it to the instructor in class by the deadline. Failure to follow the instructions exactly will incur a 10% penalty on the grade for this assignment. Please submit your assignment to your github repository as a SINGLE ZIP FILE with the following format: last name_firstname_bankers.zip, ALL IN LOWER CASE.



```
public interface Bank {
    public void addCustomer(int threadNum, int[] maxDemand, int[] allocated);    // add customer to Bank

    public void getState();    // outputs available, allocation, max, and need matrices

                               // request resources; specify number of customer being added, maxDemand for customer
                               //      returns if request is grant
    public boolean requestResources(int threadNum, int[] request);

    public void releaseResources(int threadNum, int[] release);    // release the customer's resources

    public int[] getAllocation(int threadNum);    // get allocation for given customer

    public void inactivateCustomer(int threadNum);  // mark customer as not needing resources any longer
}
```

*Bank's interface*

---

```
0,1,0,7,5,3
2,0,0,3,2,2
3,0,2,9,0,2
2,1,1,2,2,2
0,0,2,4,3,3
```

### Infile.txt
*for each process, this file lists the resources currently allocated and the maximum resources needed*
*e.g., P0 has 0,1,0 of R0,R1,R2, and needs a maximum of 7,5,3 of each of those*

```
adding customer 0...
adding customer 1...
adding customer 2...
adding customer 3...
adding customer 4...

Enter * to get state of system, OR Enter <RQ | RL> <customer number> <resource #0> <#1> <#2> ...
*
available [10 5 7]
        ALLOCATED    MAXIMUM      NEED
        [0 1 0]      [7 5 3]      [7 4 3]
        [2 0 0]      [3 2 2]      [1 2 2]
        [3 0 2]      [9 0 2]      [6 0 0]
        [2 1 1]      [2 2 2]      [0 1 1]
        [0 0 2]      [4 3 3]      [4 3 1]

Enter * to get state of system, OR Enter <RQ | RL> <customer number> <resource #0> <#1> <#2> ...
RQ 1 99 1 1
*99*
*1*
*1*
requesting resources: [99 1 1] for customer: 1...

#P1 RQ:[99 1 1],needs:[1 2 2],available=[10 5 7] DENIED
...request DENIED
```
*Interactive mode*

```java
public class Customer implements Runnable {
    public static final int COUNT = 5;    // number of threads

    private int numOfResources;    // N different resources
    private int[] maxDemand;       // maximum this thread will demand
    private int customerNum;       // customer number
    private int[] request;         // request it is making

    private java.util.Random rand; // random number generator

    private Bank theBank;          // synchronizing object
```
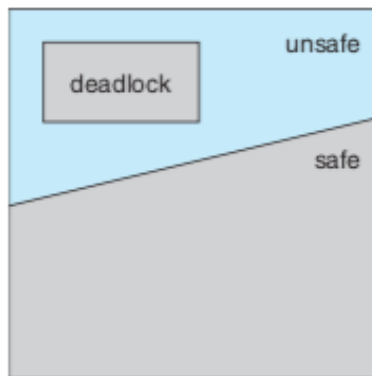*Customer (a process): Runnable in Java*

```
Enter * to get state of system, OR Enter <RQ | RL> <customer number> <resource #0> <#1> <#2> ...
RQ 1 1 1 1
*1*
*1*
*1*
requesting resources: [1 1 1] for customer: 1...

#P1 RQ:[1 1 1],needs:[1 2 2],available=[10 5 7]   ---> APPROVED, #P1 now at:[3 1 1]
available [9 4 6]
        ALLOCATED    MAXIMUM      NEED
        [0 1 0]      [7 5 3]      [7 4 3]
        [3 1 1]      [3 2 2]      [0 1 1]
        [3 0 2]      [9 0 2]      [6 0 0]
        [2 1 1]      [2 2 2]      [0 1 1]
        [0 0 2]      [4 3 3]      [4 3 1]
...request Approved
```
*Another example of **Interactive mode***


unsafe
deadlock
safe

```
adding customer 0...
adding customer 1...
adding customer 2...
adding customer 3...
adding customer 4...
FACTORY: created threads
#P0 RQ:[3 4 0],needs:[7 4 3],available=[10 5 7]FACTORY: started threads
    ---> APPROVED, #P0 now at:[3 5 0]
available [7 1 7]
        ALLOCATED    MAXIMUM      NEED
        [3 5 0]      [7 5 3]      [4 0 3]
        [2 0 0]      [3 2 2]      [1 2 2]
        [3 0 2]      [9 0 2]      [6 0 0]
        [2 1 1]      [2 2 2]      [0 1 1]
        [0 0 2]      [4 3 3]      [4 3 1]
#P4 RQ:[1 2 1],needs:[4 3 1],available=[7 1 7] DENIED
#P3 RQ:[0 0 1],needs:[0 1 1],available=[7 1 7]   ---> APPROVED, #P3 now at:[2 1 2]
available [7 1 6]
        ALLOCATED    MAXIMUM      NEED
        [3 5 0]      [7 5 3]      [4 0 3]
        [2 0 0]      [3 2 2]      [1 2 2]
        [3 0 2]      [9 0 2]      [6 0 0]
        [2 1 2]      [2 2 2]      [0 1 0]
        [0 0 2]      [4 3 3]      [4 3 1]
#P4 RQ:[1 3 0],needs:[4 3 1],available=[7 1 6] DENIED
#P2 RQ:[6 0 0],needs:[6 0 0],available=[7 1 6]   ---> APPROVED, #P2 now at:[9 0 2]
available [1 1 6]
        ALLOCATED    MAXIMUM      NEED
        [3 5 0]      [7 5 3]      [4 0 3]
        [2 0 0]      [3 2 2]      [1 2 2]
        [9 0 2]      [9 0 2]      [0 0 0]
        [2 1 2]      [2 2 2]      [0 1 0]
        [0 0 2]      [4 3 3]      [4 3 1]
```
*Automatic mode at start of simulation – 5 processes (customers)*

R1 R2 R3 — Claim matrix C / Allocation matrix A / C – A

(a) Initial state

| | R1 | R2 | R3 | | R1 | R2 | R3 | | R1 | R2 | R3 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 3 | 2 | 2 | P1 | 1 | 0 | 0 | P1 | 2 | 2 | 2 |
| | 6 | 1 | 3 | P2 | 6 | 1 | 2 | P2 | 0 | 0 | 1 |
| | 3 | 1 | 4 | P3 | 2 | 1 | 1 | P3 | 1 | 0 | 3 |
| | 4 | 2 | 2 | P4 | 0 | 0 | 2 | P4 | 4 | 2 | 0 |

Resource vector R: 9 3 6  Available vector V: 0 1 1

(b) P2 runs to completion

| | R1 | R2 | R3 | | R1 | R2 | R3 | | R1 | R2 | R3 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 3 | 2 | 2 | P1 | 1 | 0 | 0 | P1 | 2 | 2 | 2 |
| | 0 | 0 | 0 | P2 | 0 | 0 | 0 | P2 | 0 | 0 | 0 |
| | 3 | 1 | 4 | P3 | 2 | 1 | 1 | P3 | 1 | 0 | 3 |
| | 4 | 2 | 2 | P4 | 0 | 0 | 2 | P4 | 4 | 2 | 0 |

Resource vector R: 9 3 6  Available vector V: 6 2 3

*(from Stallings, OS, ch. 6)*

```
#P3 RQ:[0 1 0],needs:[0 1 0],available=[1 1 6]   ---> APPROVED, #P3 now at:[2 2 2]
available [1 0 6]
      ALLOCATED   MAXIMUM      NEED
      [3 5 0]     [7 5 3]     [4 0 3]
      [2 0 0]     [3 2 2]     [1 2 2]
      [9 0 2]     [9 0 2]     [0 0 0]
      [2 2 2]     [2 2 2]     [0 0 0]
      [0 0 2]     [4 3 3]     [4 3 1]
-----------------------------> #P3 has all its resources!  RELEASING ALL and SHUTTING DOWN...

======================= customer #3 releasing:[2 2 2],allocated=[0 0 0]
#P1 RQ:[0 0 1],needs:[1 2 2],available=[3 2 8]   ---> APPROVED, #P1 now at:[2 0 1]
available [3 2 7]
      ALLOCATED   MAXIMUM      NEED
      [3 5 0]     [7 5 3]     [4 0 3]
      [2 0 1]     [3 2 2]     [1 2 1]
      [9 0 2]     [9 0 2]     [0 0 0]
      -------     -------     -------
      [0 0 2]     [4 3 3]     [4 3 1]
#P4 RQ:[1 0 0],needs:[4 3 1],available=[3 2 7]   ---> APPROVED, #P4 now at:[1 0 2]
available [2 2 7]
      ALLOCATED   MAXIMUM      NEED
      [3 5 0]     [7 5 3]     [4 0 3]
      [2 0 1]     [3 2 2]     [1 2 1]
      [9 0 2]     [9 0 2]     [0 0 0]
      -------     -------     -------
      [1 0 2]     [4 3 3]     [3 3 1]
#P0 RQ:[3 0 3],needs:[4 0 3],available=[2 2 7] DENIED
#P4 RQ:[1 0 1],needs:[3 3 1],available=[2 2 7]   ---> APPROVED, #P4 now at:[2 0 3]
available [1 2 6]
      ALLOCATED   MAXIMUM      NEED
      [3 5 0]     [7 5 3]     [4 0 3]
      [2 0 1]     [3 2 2]     [1 2 1]
      [9 0 2]     [9 0 2]     [0 0 0]
      -------     -------     -------
      [2 0 3]     [4 3 3]     [2 3 0]
-----------------------------> #P2 has all its resources!  RELEASING ALL and SHUTTING DOWN...
```

P3 has released all of its resources, and shut down



(c) P1 runs to completion

| | R1 | R2 | R3 | | R1 | R2 | R3 | | R1 | R2 | R3 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | 0 | 0 | 0 | P1 | 0 | 0 | 0 | P1 | 0 | 0 | 0 |
| P2 | 0 | 0 | 0 | P2 | 0 | 0 | 0 | P2 | 0 | 0 | 0 |
| P3 | 3 | 1 | 4 | P3 | 2 | 1 | 1 | P3 | 1 | 0 | 3 |
| P4 | 4 | 2 | 2 | P4 | 0 | 0 | 2 | P4 | 4 | 2 | 0 |

Resource vector R: 9 3 6  Available vector V: 7 2 3

(d) P3 runs to completion

| | R1 | R2 | R3 | | R1 | R2 | R3 | | R1 | R2 | R3 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | 0 | 0 | 0 | P1 | 0 | 0 | 0 | P1 | 0 | 0 | 0 |
| P2 | 0 | 0 | 0 | P2 | 0 | 0 | 0 | P2 | 0 | 0 | 0 |
| P3 | 0 | 0 | 0 | P3 | 0 | 0 | 0 | P3 | 0 | 0 | 0 |
| P4 | 4 | 2 | 2 | P4 | 0 | 0 | 2 | P4 | 4 | 2 | 0 |

Resource vector R: 9 3 6  Available vector V: 9 3 4

*(continued from Stallings, OS, ch. 6)*

```
======================= customer #2 releasing:[9 0 2],allocated=[0 0 0]
#P1 RQ:[0 2 1],needs:[1 2 1],available=[10 2 8]   ---> APPROVED, #P1 now at:[2 2 2]
available [10 0 7]
      ALLOCATED   MAXIMUM      NEED
      [3 5 0]     [7 5 3]     [4 0 3]
      [2 2 2]     [3 2 2]     [1 0 0]
      -------     -------     -------
      -------     -------     -------
      [2 0 3]     [4 3 3]     [2 3 0]
#P0 RQ:[3 0 0],needs:[4 0 3],available=[10 0 7]   ---> APPROVED, #P0 now at:[6 5 0]
available [7 0 7]
      ALLOCATED   MAXIMUM      NEED
      [6 5 0]     [7 5 3]     [1 0 3]
      [2 2 2]     [3 2 2]     [1 0 0]
      -------     -------     -------
      -------     -------     -------
      [2 0 3]     [4 3 3]     [2 3 0]
#P4 RQ:[1 1 0],needs:[2 3 0],available=[7 0 7] DENIED
#P4 RQ:[2 0 0],needs:[2 3 0],available=[7 0 7]   ---> APPROVED, #P4 now at:[4 0 3]
available [5 0 7]
      ALLOCATED   MAXIMUM      NEED
      [6 5 0]     [7 5 3]     [1 0 3]
      [2 2 2]     [3 2 2]     [1 0 0]
      -------     -------     -------
      -------     -------     -------
      [4 0 3]     [4 3 3]     [0 3 0]
#P0 RQ:[0 0 1],needs:[1 0 3],available=[5 0 7]   ---> APPROVED, #P0 now at:[6 5 1]
available [5 0 6]
      ALLOCATED   MAXIMUM      NEED
      [6 5 1]     [7 5 3]     [1 0 2]
      [2 2 2]     [3 2 2]     [1 0 0]
      -------     -------     -------
      -------     -------     -------
      [4 0 3]     [4 3 3]     [0 3 0]
```

P2 has also shut down; several allocations approved for P0 and P4

R1 R2 R3

|  | R1 | R2 | R3 |
|---|---|---|---|
| P1 | 3 | 2 | 2 |
| P2 | 6 | 1 | 3 |
| P3 | 3 | 1 | 4 |
| P4 | 4 | 2 | 2 |

Claim matrix C

|  | R1 | R2 | R3 |
|---|---|---|---|
| P1 | 1 | 0 | 0 |
| P2 | 5 | 1 | 1 |
| P3 | 2 | 1 | 1 |
| P4 | 0 | 0 | 2 |

Allocation matrix A

|  | R1 | R2 | R3 |
|---|---|---|---|
| P1 | 2 | 2 | 2 |
| P2 | 1 | 0 | 2 |
| P3 | 1 | 0 | 3 |
| P4 | 4 | 2 | 0 |

C – A

| R1 | R2 | R3 |
|---|---|---|
| 9 | 3 | 6 |

Resource vector R

| R1 | R2 | R3 |
|---|---|---|
| 1 | 1 | 2 |

Available vector V

(a) Initial state

|  | R1 | R2 | R3 |
|---|---|---|---|
| P1 | 3 | 2 | 2 |
| P2 | 6 | 1 | 3 |
| P3 | 3 | 1 | 4 |
| P4 | 4 | 2 | 2 |

Claim matrix C

|  | R1 | R2 | R3 |
|---|---|---|---|
| P1 | 2 | 0 | 1 |
| P2 | 5 | 1 | 1 |
| P3 | 2 | 1 | 1 |
| P4 | 0 | 0 | 2 |

Allocation matrix A

|  | R1 | R2 | R3 |
|---|---|---|---|
| P1 | 1 | 2 | 1 |
| P2 | 1 | 0 | 2 |
| P3 | 1 | 0 | 3 |
| P4 | 4 | 2 | 0 |

C – A

| R1 | R2 | R3 |
|---|---|---|
| 9 | 3 | 6 |

Resource vector R

| R1 | R2 | R3 |
|---|---|---|
| 0 | 1 | 1 |

Available vector V

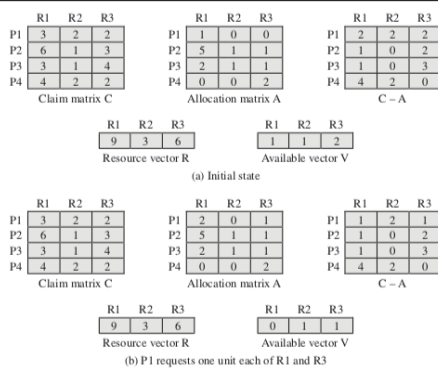(b) P1 requests one unit each of R1 and R3

**Figure 6.8  Determination of an Unsafe State**

*(from Stallings, OS, ch. 6)*

```
#P1 RQ:[1 0 0],needs:[1 0 0],available=[5 0 6]   ---> APPROVED, #P1 now at:[3 2 2]
available [4 0 6]
        ALLOCATED    MAXIMUM      NEED
        [6 5 1]      [7 5 3]      [1 0 2]
        [3 2 2]      [3 2 2]      [0 0 0]
        -------      -------      -------
        -------      -------      -------
        [4 0 3]      [4 3 3]      [0 3 0]
-------------------------------> #P1 has all its resources!  RELEASING ALL and SHUTTING DOWN...

=========================== customer #1 releasing:[3 2 2],allocated=[0 0 0]
#P4 RQ:[0 3 0],needs:[0 3 0],available=[7 2 8] DENIED
#P4 RQ:[0 1 0],needs:[0 3 0],available=[7 2 8]   ---> APPROVED, #P4 now at:[4 1 3]
available [7 1 8]
        ALLOCATED    MAXIMUM      NEED
        [6 5 1]      [7 5 3]      [1 0 2]
        -------      -------      -------
        -------      -------      -------
        -------      -------      -------
        [4 1 3]      [4 3 3]      [0 2 0]
#P0 RQ:[1 0 0],needs:[1 0 2],available=[7 1 8]   ---> APPROVED, #P0 now at:[7 5 1]
available [6 1 8]
        ALLOCATED    MAXIMUM      NEED
        [7 5 1]      [7 5 3]      [0 0 2]
        -------      -------      -------
        -------      -------      -------
        -------      -------      -------
        [4 1 3]      [4 3 3]      [0 2 0]
#P4 RQ:[0 2 0],needs:[0 2 0],available=[6 1 8] DENIED
#P0 RQ:[0 0 1],needs:[0 0 2],available=[6 1 8]   ---> APPROVED, #P0 now at:[7 5 2]
available [6 1 7]
        ALLOCATED    MAXIMUM      NEED
        [7 5 2]      [7 5 3]      [0 0 1]
        -------      -------      -------
        -------      -------      -------
        -------      -------      -------
        [4 1 3]      [4 3 3]      [0 2 0]
#P4 RQ:[0 2 0],needs:[0 2 0],available=[6 1 7] DENIED
```

*P1 has released its resources, then shut down*
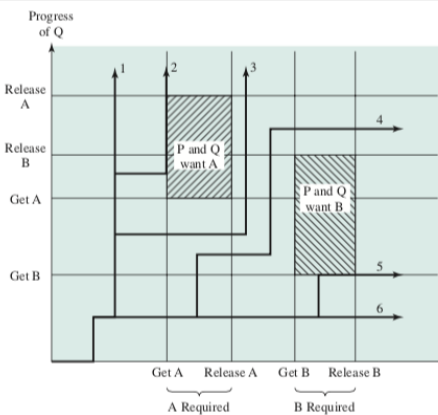


*Getting, then releasing resources, is a path to avoiding deadlock*

```
#P4 RQ:[0 1 0],needs:[0 2 0],available=[6 1 7]   ---> APPROVED, #P4 now at:[4 2 3]
available [6 0 7]
        ALLOCATED    MAXIMUM      NEED
        [7 5 2]      [7 5 3]      [0 0 1]
        -------      -------      -------
        -------      -------      -------
        -------      -------      -------
        [4 2 3]      [4 3 3]      [0 1 0]
#P0 RQ:[0 0 1],needs:[0 0 1],available=[6 0 7]   ---> APPROVED, #P0 now at:[7 5 3]
available [6 0 6]
        ALLOCATED    MAXIMUM      NEED
        [7 5 3]      [7 5 3]      [0 0 0]
        -------      -------      -------
        -------      -------      -------
        -------      -------      -------
        [4 2 3]      [4 3 3]      [0 1 0]
#P4 RQ:[0 1 0],needs:[0 1 0],available=[6 0 6] DENIED
#P4 RQ:[0 1 0],needs:[0 1 0],available=[6 0 6] DENIED
#P4 RQ:[0 1 0],needs:[0 1 0],available=[6 0 6] DENIED
-------------------------------> #P0 has all its resources!  RELEASING ALL and SHUTTING DOWN...

=========================== customer #0 releasing:[7 5 3],allocated=[0 0 0]
#P4 RQ:[0 1 0],needs:[0 1 0],available=[13 5 9]   ---> APPROVED, #P4 now at:[4 3 3]
available [13 4 9]
        ALLOCATED    MAXIMUM      NEED
        -------      -------      -------
        -------      -------      -------
        -------      -------      -------
        -------      -------      -------
        [4 3 3]      [4 3 3]      [0 0 0]
-------------------------------> #P4 has all its resources!  RELEASING ALL and SHUTTING DOWN...

=========================== customer #4 releasing:[4 3 3],allocated=[0 0 0]

Process finished with exit code 0
```

*All processes have finished*

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

**Your name and company name:** _____

**Repository** `https://github.com/`_____/_____

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: <br> *  OR  **<Rq \| Rl> <process number> <R0 resources> <R1 resources> …** <br> **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. <br> Increasing resource types _____ <br> Increasing processes _____ |
| ❏ | ❏ | Tested the simulation to find the most likely combination of processes and resources to cause deadlock _____ |
| ❏ | ❏ | 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

_____

_____

_____