Group Project 3: All about deadlock

CECS 326 – Operating Systems

You should submit the required deliverable materials on Canvas by **11:59pm, March 31st, (Sunday), 2024.**

1. Problem Description

For this project, you will write a program that implements the banker's algorithm discussed in the slides. Customers request and release resources from the bank. The banker will grant a request only if it leaves the system in a safe state. A request that leaves the system in an unsafe state will be denied. Although the code examples that describe this project are illustrated in C, you may also develop a solution using Java.

The Banker

The banker will consider requests from *n* customers for *m* resources types, as outlined in slides. The banker will keep track of the resources using the following data structures:

```
#define NUMBER_OF_CUSTOMERS 5
#define NUMBER_OF_RESOURCES 4

/* the available amount of each resource */
int available[NUMBER_OF_RESOURCES];

/*the maximum demand of each customer */
int maximum[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];

/* the amount currently allocated to each customer */
int allocation[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];

/* the remaining need of each customer */
int need[NUMBER_OF_CUSTOMERS][NUMBER_OF_RESOURCES];
```

The banker will grant a request if it satisfies the safety algorithm. If a request does not leave the system in a safe state, the banker will deny it. Function prototypes for requesting and releasing resources are as follows:

```
int request_resources(int customer_num, int request[]);

void release_resources(int customer_num, int release[]);
```

The request *resources()* function should return 0 if successful and –1 if unsuccessful.

Testing Your Implementation

Design a program that allows the user to interactively enter a request for resources, to release resources, or to output the values of the different data structures (*available, maximum, allocation, and need*) used with the banker's algorithm.

You should invoke your program by passing the number of resources of each type on the command line. For example, if there were four resource types, with ten instances of the first type, five of the second type, seven of the third type, and eight of the fourth type, you would invoke your program as follows:

./xxx.out 10 5 7 8

The available array would be initialized to these values. Your program will initially read in a file containing the maximum number of requests for each customer. For example, if there are five customers and four resources, the input file would appear as follows:

6,4,7,3
4,2,3,2
2,5,3,3
6,3,3,2
5,6,7,5

where each line in the input file represents the maximum request of each resource type for each customer. Your program will initialize the maximum array to these values.

Your program will then have the user enter commands responding to a request of resources, a release of resources, or the current values of the different data structures. Use the command 'RQ' for requesting resources, 'RL' for releasing resources, and '*' to output the values of the different data structures. For example, if customer 0 were to request the resources (3, 1, 2, 1), the following command would be entered:

RQ 0 3 1 2 1

Your program would then output whether the request would be satisfied or denied using the safety algorithm.

Similarly, if customer 4 were to release the resources (1, 2, 3, 1), the user would enter the following command:

RL 4 1 2 3 1

Finally, if the command '*' is entered, your program would output the values of the *available*, *maximum*, *allocation*, and *need* arrays.

2. The Required Deliverable Materials

(1) A README file, which describes how we can compile and run your code.
(2) Your source code, should submit in the required format.
(3) Your short report, which discusses the design of your program.
(4) A recorded video shows the output and runtime

3. Submission Requirements

You need to strictly follow the instructions listed below:

1) This is a **group project**, please submit a .zip/.rar file that contains all files, only one submission from one group.

2) Make a **video** to record your code execution and outputs. The video should present your name or time as identification (You are suggested to upload the video to YouTube and put the link into your report).

3) The submission should include your **source code** and **project report**. Do not submit your binary code. Project report should contain your groupmates name and ID.

4) Your code must **be able to compile**; otherwise, you will receive a grade of zero.

5) Your code should not produce anything else other than the required information in the output file.

7) If you code is **partially completed**, please explain the details in the report what has been completed and the status of the missing parts, we will grade it based on the entire performance.

8) Provide **sufficient comments** in your code to help the TA understand your code. This is important for you to get at least partial credit in case your submitted code does not work properly.

Grading criteria:

Details	Points
Have a README file shows how to compile and test your submission	5 pts
Submitted code has proper comments to show the design	15 pts
Screen a ***video*** to record code execution and outputs	10 pts
Have a **report** (pdf or word) file explains the details of your entire design	20 pts
Report contains clearly individual contributions of your group mates	5 pts
Code can be compiled and shows correct outputs	45 pts

4. Policies

1) Late submissions will be graded based on our policy discussed in the course syllabus.
2) Code-level discussion is **prohibited**. We will use anti-plagiarism tools to detect violations of this policy.