CSC 4320: Operating Systems

50 Points

Homework 5

© Instructor: Dr. Md. Mahfuzur Rahman

Spring 2025

Work on the following problems and submit your own answers. You are allowed to discuss with other students. However, do not copy the solutions from peers or other sources. If the assignment has programming component, your program(s) must compile with **gcc** and execute on **snowball.cs.gsu.edu**! Please see https://cscit.cs.gsu.edu/sp/guide/snowball for more details.

Instructions:

- Upload an electronic copy (MS word or pdf) of your answer sheet to the folder named "HW5" in iCollege.
- Please add the course number, homework number, and your name at the top of your answer sheet.
- Please write down your answers with the question number only in the answer sheet.
- Also submit your .c file (c program), if you are asked to write a program.
- Name your file in the format of CSC4320_HW5_FisrtnameLastname (.docx/.pdf)
- Deadline: Submit by April 11, 2025, 11:59 pm

1. (5 points) Suppose that a system is in an unsafe state. Show that it is possible for the threads to complete their execution without entering a deadlocked state. Show by an example.

2. Consider the following snapshot of a system:

	Allocation				**Max**			
	A	**B**	**C**	**D**	**A**	**B**	**C**	**D**
T_0	3	0	1	4	5	1	1	7
T_1	2	2	1	0	3	2	1	1
T_2	3	1	2	1	3	3	2	1
T_3	0	5	1	0	4	6	1	2
T_4	4	2	1	2	6	3	2	5

Using the banker's algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the threads may complete. Otherwise, illustrate why the state is unsafe.

(a) (5 points) `Available` = (0,3,0,1)

(b) (5 points) `Available` = (1,0,0,2)

3. (15 points) Which of the six resource-allocation graphs shown below illustrate deadlock? For those situations that are deadlocked, provide the cycle of threads and resources. Where there is not a deadlock situation, illustrate the order in which the threads may complete execution.

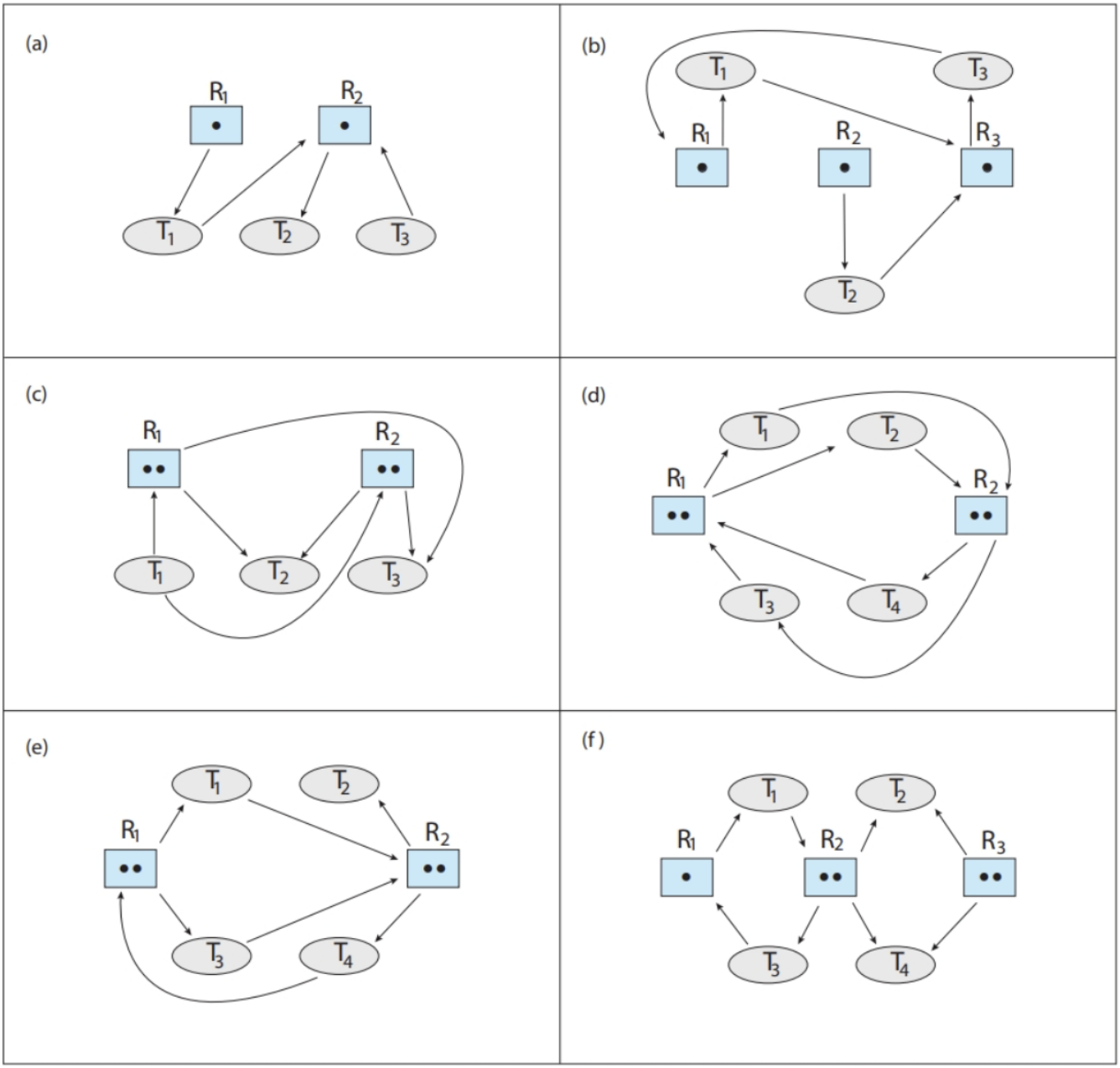

Figure 1: Resource Allocation Graph.

4. (20 points) **Implement Banker's Algorithm:** For this problem, you will write a program that implements the banker's algorithm as discussed in Chapter 8. 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. You are expected to write your solution in C.

   The Banker: The banker will consider requests from n customers for m resources types. 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 (call it `banker.c`) 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:

`./banker 10 5 7 8` `banker` is your executable program after compilation

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.

Question:	1	2	3	4	Total
Points:	5	10	15	20	50
Bonus Points:	0	0	0	0	0
Score:					