8.13 Programming projects

Banker's algorithm

For this project, you will write a program that implements the banker's algorithm discussed in Section Banker's algorithm. 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 Section Banker's algorithm. 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 outlined in Section 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:

```
./a.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 outlined in Section 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.

How was this

section? **Provide section feedback**