Lab 4 - Parallelism Synchronization

Team Information

Section: PRA0102

Chirag Sethi (1006219263)

Yuhe Chen (1005689480)

Report

Design

```
/** Design:
 * The program uses a class `Teller` to represent each banking teller and a class Customer to represent each banking customer.
 * The program uses a class `Semaphore` to implement the Semaphore mechanism for controlling access to the tellers.
 * The program uses a class `BankingSystem` to manage the tellers and customers and to serve the customers.
 * The program initializes the tellers and adds the customers to the system.
 * The program uses a thread to serve the customers.
 * The program uses a queue to store the customers and a vector to store the tellers.
 * The program uses a Semaphore to control access to the tellers.
 * The program generates a random service time for each customer and waits for the service time before releasing the teller.
 */
```

Measure the number of threads in your program with their life time (from the time being created until the time being detached) and save the results in a table

	Thread 1	Thread 2	Thread 3	Thread 4	Thread 5	Thread 6	Thread 7	Thread 8	Thread 9	Thread 10
Start	0	0	0	4.542	4.58	4.594	5.676	7.118	7.315	8.434
Duration	4.542	4.58	4.594	2.576	1.096	2.721	3.376	4.73	1.119	2.649
Finish	4.542	4.58	4.594	7.118	5.676	7.315	9.052	11.848	8.434	11.083

Measure the execution time of your program (the whole program) by changing the number of threads; Run your program 5 times and find the average among the execution times and save the results in a table

Average execution time from 5 runs with 3 maximum parallel threads is 11.02.

If we reduce the number of parallel threads (number of Tellers), the execution time increases.

If we increase the number of parallel threads (number of Tellers), the execution time decreases.

Small test cases and their sample outputs, representing how your program works.

You need to make a queue of requests trying to have access to the shared resources (as test inputs). It is recommended to make two scenarios or two queues so that in one of them all requests can get access to the shared resources properly but in the other one, a request in the queue has to wait exceeding the limit time due to the service being provided to another request and it was generated randomly, and so make n error message for the system).

In this case, 3 tellers are serving 10 customers with customer id from 1 to 10.

```
Semaphore initialized to 3
Teller 1: Customer 1
        Starts  : 0
        Duration: 4.542
        Ends    : 4.542
Teller 2: Customer 2
```

```
        Starts  : 0
        Duration: 4.58
        Ends    : 4.58
Teller 3: Customer 3
        Starts  : 0
        Duration: 4.594
        Ends    : 4.594
Teller 1: Customer 4
        Starts  : 4.542
        Duration: 2.576
        Ends    : 7.118
Teller 2: Customer 5
        Starts  : 4.58
        Duration: 1.096
        Ends    : 5.676
Teller 3: Customer 6
        Starts  : 4.594
        Duration: 2.721
        Ends    : 7.315
Teller 2: Customer 7
        Starts  : 5.676
        Duration: 3.376
        Ends    : 9.052
Teller 1: Customer 8
        Starts  : 7.118
        Duration: 4.73
        Ends    : 11.848
Teller 3: Customer 9
        Starts  : 7.315
        Duration: 1.119
        Ends    : 8.434
Teller 3: Customer 10
        Starts  : 8.434
        Duration: 2.649
        Ends    : 11.083
```

We can verify the correctness of our program through output:

> For example, when all 3 tellers are servicing customers at the very beginning. The first teller to finish (Teller 1) start to serve customer 4. Then the second to finish (Teller 2) starts to server customer 5. It proceed in this order until all customers are served.

Make a proper function to display the queue situation as well as the shared resources situation at the same time, for example after each service.

`displayQueueResource()` is used to display queue situation as well as the shared resources. An example call to `displayQueueResource()` at the beginning of the program gives the following output:

```
Teller 1 is available
Teller 2 is available
Teller 3 is available
Number of available tellers: 3
Number of customers in queue: 10
```

Full source codes with proper comments at proper points, representing each class and function definition

Please see comments in the submitted file.

Discussion about which concurrent program features are created, used and deleted in the flow of your program

The 3 tellers are serving customers concurrently.

It's created using thread creation by `thread t()`.

`t.detach()` will make the thread detached, so that when the thread finished execution, its resources will automatically be recycled. The concurrency is "deleted" this way.

A readme file of how to run and test your program completely

```
How to run and test your program completely:
Compile: `make`
Run: ./lab4
```

```
The results can be verified through the output.
```