COMP3520 Operating Systems Internals Assignment 1 – Synchronization

General Instructions

This assignment is about synchronization. It consists of two **compulsory** tasks:

1. Write a multithreaded Pthread program to solve the synchronization problem as described in the section "Stylish Hairdressing Salon"; and
2. Answer the assigned discussion document questions (which are provided in a separate document).

This assignment is an individual assignment. Whilst you are permitted to discuss this assignment with other students, the work that you submit must be your own work. You should not incorporate the work of other students (past or present) or Artificial Intelligence (AI) tools into your source code or discussion document.

You will be required to submit your discussion document to *Turnitin* for similarity checking as part of assignment submission. The examiner may use other similarity checking tools in addition to *Turnitin*. Your source code may also be checked.

Submit your source code and report to the appropriate submission inboxes in the COMP3520 Canvas website.

The Problem: Stylish Hairdressing Salon

In Stylish Hairdressing Salon, there are K barbers. Each time, a barber takes a minimum of T1 units of time and a maximum of T2 units of time to service one customer.

The salon has N waiting chairs. When a customer arrives and if there are free waiting chairs, he/she takes a ticket and then sits on a waiting chair, waiting for their ticket number to be called. If all the waiting chairs are occupied, however, the newly arrived customer simply leaves (and vanishes). The tickets are numbered from 1 to N and initially placed in ascending order. Customers always take a ticket from one end and return the ticket to the other end after being called. In this way, the ticket numbers are always maintained in a perfect cyclic order.

There is one shop assistant looking after the customers. When a barber becomes available, the assistant will call one waiting customer to get haircut by that barber. Customers will be called in a first-come-first-served order. Note that customers are not allowed to choose barbers in this salon.

This is a synchronization problem. To solve this problem, you need to consider various situations. For example,

- when there are no customers and all barbers are waiting;
- when all barbers are busy and customers are waiting; and
- how to assign customers to a specific barber when that barber is available.

In your implementation, only barbers determine how long it takes to service each customer. The program will terminate only after all barbers have done their work. You need to write three routines, i.e., *customer*, *barber* and *assistant*, in addition to *main*.

To implement synchronization between various threads, you may use Pthread mutexes, and condition variables. **However, you must not use any other type of synchronization mechanism.**

Your program needs to ask the user to enter the following parameters:

- *M*: the total number of customers;
- *K*: the number of barbers working in the salon;
- N: the total number of waiting chairs in the salon;
- *T1*: the minimum number of units of time for a barber to service a customer;
- *T2*: the maximum number of units of time for a barber to service a customer;
- *T3*: the minimum number of units of time between two customer arrivals; and
- *T4*: the maximum number of units of time between two customer arrivals.

To check whether your program functions properly, your program must print the statements described below.

Each **Customer thread** must print the following status messages wherever appropriate:

- "Customer [*id*]: I have arrived at the barber shop." – New customer with customer *id* arrives
- "Customer [*id*]: Oh no! All seats have been taken and I'll leave now!" – All waiting chairs are occupied and the customer leaves
- "Customer [*id*]: I'm lucky to get a free seat and a ticket numbered *n*" – Take a ticket numbered *n* and sit on a free waiting chair
- "Customer [*id*]: My ticket number *n* has been called. Hello, Barber [*id*]." – To get haircut next by Barber [*id*]
- "Customer [*id*]: Well done. Thanks Barber [*id*]. Bye!" – Get haircut done and leave

Each **Barber thread** must print the following status messages wherever appropriate:

- "Barber [*id*]: I'm now ready to accept a customer." – Wait for a new customer
- "Barber [*id*]: Hello, Customer *n*." – To service the assigned customer with a ticket number *n*
- "Barber [*id*]: Finished cutting. Good bye Customer *n*." – Finished servicing Customer *n*
- "Barber [*id*]: Thanks Assistant. See you tomorrow!" – Finished working for the day and leaves

The **Assistant thread** must print the following status messages wherever appropriate:

- "Assistant: I'm waiting for customers." – Wait for customers
- "Assistant: I'm waiting for a barber to become available." – Wait for barber
- "Assistant: Assign Customer *n* to Barber [*id*]." – Assign Customer *n* to Barber [*id*]
- "Assistant: Hi barbers. We've finished the work for the day. See you all tomorrow!" – Call for all barbers to finish and leave

Note that in the above messages, Customer *n* is the customer with a ticket numbered *n*, and the ticket number *n* may not be equal to the customer's [*id*].

When all customers have been serviced, the main thread must print the following status message and then terminate:

- "Main thread: All customers have now been served. Salon is closed now."

Additional Requirements

Source Code

Your solution must be implemented in the C language.

Your source code needs to be properly commented and appropriately structured to allow another programmer who has a working knowledge of C to understand and easily maintain your code.

You need to include a well-structured and properly commented *makefile* that allows for the compilation of your source code using the *make* command on the School of Computer Science servers.

Testing and Debugging

You are responsible for testing and debugging your source code.

It is crucial that you ensure that the source code you submit compiles correctly on the School of Computer Science servers and that the resulting binary functions as intended. If you submit source code that cannot be compiled on the School servers, marks will be heavily deducted.

Discussion Document

You are required to answer all assigned questions in a separate written document. The questions and requirements specific to the discussion document will be provided in a separate document.

Other Matters

Marking criteria for the source code and discussion document will be provided separately.

To maximize your chances of realizing your full potential in COMP3520, **please start work on this assignment promptly.**