1 Introduction

1.1 Assignment Rules

1. This assignment is meant to be solved individually and not in groups. That means the assignment is evaluated/graded individually.
2. When you believe your assignment is ready to be evaluated/graded, upload your solution using Moodle's submission system, and then it will get assessed after some days by us. Please refer to 0 for the submission structure.

2 Description

2.1 Task 1

Create two threads T_A and T_B, where each thread is doing one thing:

1. T_A prints 'A' on the screen and,
2. T_B prints 'B' on the screen

Synchronize T_A and T_B using semaphores so that the sequence AB is printed 10 times on the screen.

2.2 Task 2

Create four threads T_A, T_B, T_C and T_D where each thread is doing one thing:

1. T_A prints 'A' on the screen and,
2. T_B prints 'B' on the screen and,
3. T_C prints 'C' on the screen
4. T_D prints 'D' on the screen

Synchronize the threads using semaphores so that the sequence ABACDC is printed 5 times on the screen.

2.3 Task 3

Create a thread safe message queue (FIFO) capable of handling a single receiver and multiple senders. Thread safe means that you must protect shared data from being preempted.

The message queue shall implement the following interface:

```
public interface IMessageQueue {

    /**
      * Send a message.
      * @return true if successful, otherwise false.
      */
     boolean Send(char msg);

     /**
      * Receive a message.
      * If queue has no message, Recv() will block until one arrives.
      * @return A message.
      */
     char Recv();
```

Your class implementing this interface might have other methods as well!

Note!:

1. As an internal buffer, you are only allowed to use primitive arrays (i.e., not ArrayList, etc.). This also means that to fully utilize the buffer, a circular queue must be used.
2. Max internal buffer size is 5.
3. You must use a counting semaphore for managing the internal buffer (has messages and is full). View array elements as resources!

Instructions for how to run code in Main():

Create three senders S_A, S_B and S_C where each sender is doing the following:

1. S_A adds a 'A' to the queue,
2. S_B adds a 'B' to the queue,
3. S_C adds a 'C' to the queue

Each sender is running in its own thread and attempts to add/send messages the queue.

Create a receiver that is also running in its own thread and attempts to receive messages. Each message is printed on the screen (no newlines).

2.4 Submission structure

Your submission MUST be composed of a .zip file containing your executable source code along with the instructions for building it.