GEBZE TECHNICAL UNIVERSITY
COMPUTER ENGINEERING

CSE 437 - 2022 SPRING
REAL TIME SYSTEM ARCHITECTURES

HOMEWORK 2 REPORT

Dilara KARAKAŞ 171044010

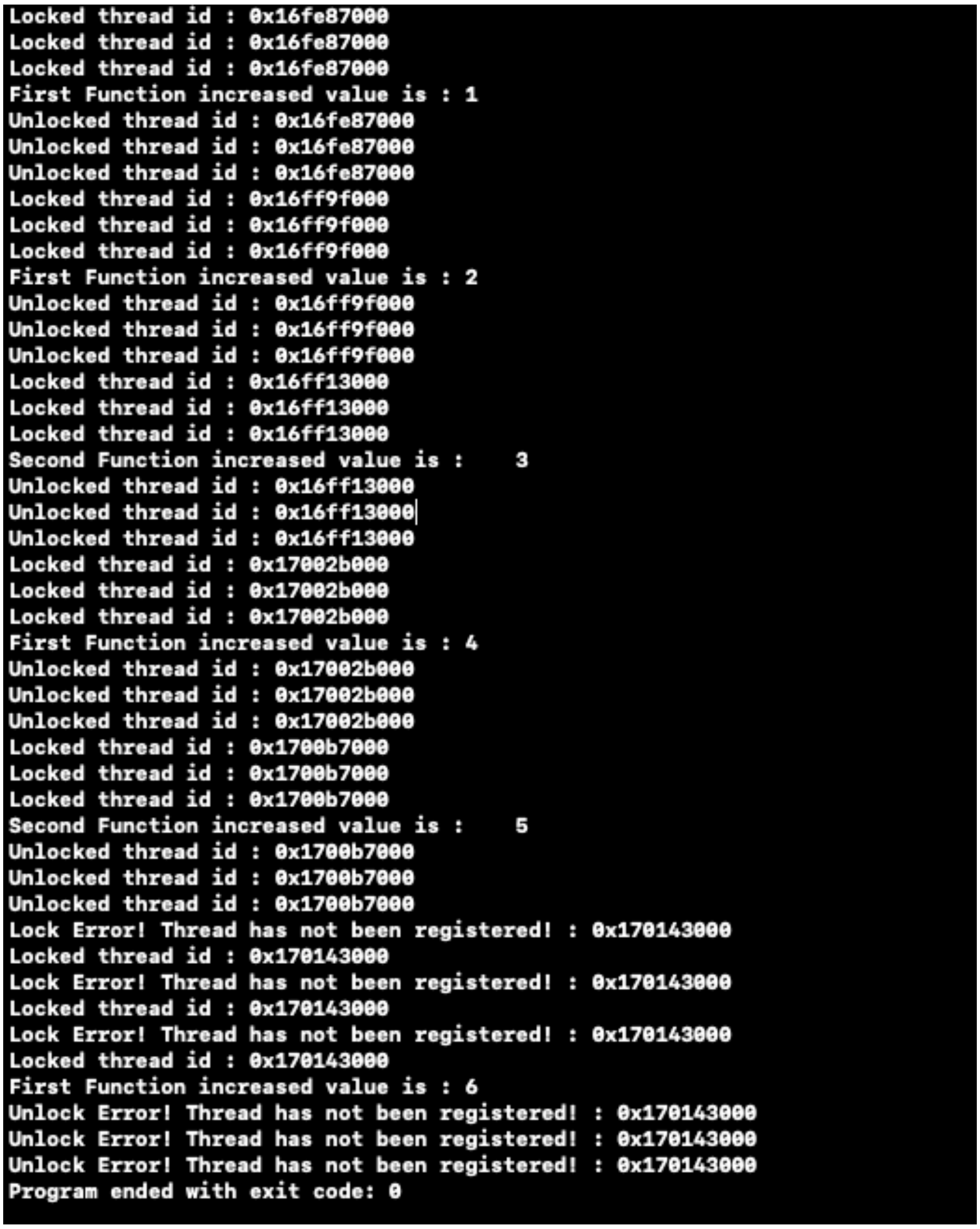
DESIGN

In this part, creating a mutex concrete class in gtu namespace which extends std::mutex concrete class. The mutex concrete class has lock() and unlock() methods to override std::mutex concrete class this methods. In gtu namespace, created a new concrete class which is called RegThreadInfo. That class used to register the created threads by users to created mutexes. The gtu::mutex concrete class keeps in private as a vector of ThreadInfo object informations as registered thread's priority, id and vector of registered mutexes. In main function, creating 5 threads as an example which uses thread 1 function() a thread 2 function() functions that doesn't have any parameter. Created threads passed to gtu::setScheduleThread() giving first parameter as thread and second parameter as thread's priority. The gtu::setScheduleThread() function sets the thread's priority correctly and registers the thread to each mutexes, used 3 mutexes in that example. After scheduling gtu::startProtocol() function must be called to start the priority ceiling protocol using conditional variable.
In thread 1 function() and thread 2 function(), locking mechanism is created using lock guard in one of them and unique lock in the other one for 3 mutexes. This functions use conditional variables to wait till register operation completed. This functions just increases a global integer variable. The lock guard and unique lock mechanism uses mutex's lock() and unlock() methods. The priority ceiling protocol is decided to be applied in these methods. In lock() method firstly checking the the current thread if it is registered to mutex if it is not has been registered yet then printing the exception message and not locking.
Otherwise checking the acquired mutex which threads have registered itself on it and the mutex has the current thread. Checking the current thread's priority comparison between acquired mutex's ceil and decide to lock or wait till acquire the mutex using conditional variable. If mutex has not been acquired then locking mechanism works and acquired reinitialized to true.

How to build

```
make
./main
```

Test



```
Locked thread id : 0x16fe87000
Locked thread id : 0x16fe87000
Locked thread id : 0x16fe87000
First Function increased value is : 1
Unlocked thread id : 0x16fe87000
Unlocked thread id : 0x16fe87000
Unlocked thread id : 0x16fe87000
Locked thread id : 0x16ff9f000
Locked thread id : 0x16ff9f000
Locked thread id : 0x16ff9f000
First Function increased value is : 2
Unlocked thread id : 0x16ff9f000
Unlocked thread id : 0x16ff9f000
Unlocked thread id : 0x16ff9f000
Locked thread id : 0x16ff13000
Locked thread id : 0x16ff13000
Locked thread id : 0x16ff13000
Second Function increased value is :    3
Unlocked thread id : 0x16ff13000
Unlocked thread id : 0x16ff13000
Unlocked thread id : 0x16ff13000
Locked thread id : 0x17002b000
Locked thread id : 0x17002b000
Locked thread id : 0x17002b000
First Function increased value is : 4
Unlocked thread id : 0x17002b000
Unlocked thread id : 0x17002b000
Unlocked thread id : 0x17002b000
Locked thread id : 0x1700b7000
Locked thread id : 0x1700b7000
Locked thread id : 0x1700b7000
Second Function increased value is :    5
Unlocked thread id : 0x1700b7000
Unlocked thread id : 0x1700b7000
Unlocked thread id : 0x1700b7000
Lock Error! Thread has not been registered! : 0x170143000
Locked thread id : 0x170143000
Lock Error! Thread has not been registered! : 0x170143000
Locked thread id : 0x170143000
Lock Error! Thread has not been registered! : 0x170143000
Locked thread id : 0x170143000
First Function increased value is : 6
Unlock Error! Thread has not been registered! : 0x170143000
Unlock Error! Thread has not been registered! : 0x170143000
Unlock Error! Thread has not been registered! : 0x170143000
Program ended with exit code: 0
```