**CSE 333 – OPERATING SYSTEMS**

**PROJECT 3**

**Group Members :** Sevda DURDU – 150115043

Emre Barkın BOZDAĞ - 150115030

Aim of this project is writing a program that uses threads and synchronization to generate a square matrix randomly created numbers. We will implement a program with four types of threads : Generate Thread, Log Thread, Mod Thread and Add Thread.

Generally, we create an inner\_matrix struct which holds 5x5 empty matrix because in this way, pass by reference the matrix is easier. We implement queue data structure for queue1(generated matrices queue) and queue2 ( for matrices which are generated by mod thread). In this queue structure queue nodes holds our inner matrix struct, id, generate status, log status, mod status and sum status.

Before starting thread functions we create simple functions for threads jobs. For example calculate\_mod, calculate\_sum, log\_into\_matrix, initialize\_matrix, create\_inner\_matrix and create\_mod\_inner\_matrix functions.

**Generate Thread:**

This thread will generate a matrix of size 5x5 and fill it with random numbers.

Firstly, we need to create a queue which nodes will hold generated matrices for this thread. But we need to create this thread only if it is not created. We enqueue all queue nodes with id number and without inner matrix. We will use that id numbers for locate inner matrices into main matrix. For this job we create global variable genf\_cond and mutex gen\_mutex1. With mutex we sure about only one thread will create queue. After creation of queue we change genf\_cond variable to prevent other threads enter same mutex.

Now, all generated threads will start to generate inner matrices. For generating matrices, thread will start to look queue nodes. If gen\_condition of node is 0, this means inner matrix is not created. We check this condition in mutex (gen\_mutex2) locks and after one thread find 0 it will change gen\_condition to 1. It means inner matrix is in generating state so other threads can’t try to generate that matrix. We also change a local variable(cond\_temp) to make generation operation outside of mutex range. So this will make generate thread work concurrently. After generation we change queue node’s gen\_status variable to 2. This means inner matrix is created so log and mod thread can use this node.

When created matrix count is equal to total inner matrix count thread will be finished.

**Log Thread:**

Log thread will get the 5x5 matrix generated by Generate Thread and allocate a bigger matrix to put multiple smaller matrices inside.

Firstly, we wait genf\_cond to be 1. This means queue1 is created. Then, one thread will create main matrix (same way of creating queue1). We check gen\_status of queue1 node and if it is created (gen\_status = 2). If we mark as it logged a matrix (we can change gen\_status to 3) and we do this operation inside locks of log\_mutex2. After mark that as logged, we locate that matrix into main matrix.

**Mod Thread:**

This Thread will get the 5x5 matrix generated by Generate Thread and find module of each number by the first number in the matrix (the number at the zeroth row and zeroth column).

Firstly, we wait creation of queue1. After that one of the mod threads will create queue2 (same way as the creation of queue1). We check gen\_status of queue1 node is bigger than 2 or equal to 2 (we can take mods of generated or logged inner matrices) and we check mod\_status of same node. If its mod matrix is not created, this will create. After creation of new matrix, we put this matrix into queue2.

**Add Thread:**

Add Thread will get the output of Mod Thread (queue2) and find a local sum of 5x5 matrixes and update a global sum.

Firstly, we wait creation of queue2. We check mod\_status and sum\_status. If mod matrix created and sum calculation is not maked thread will start to calculate. We check these conditions before operation of calculation and inside a mutex (sum\_mutex1) locks range. When we update the global sum, we make this operation inside a mutex (global\_sum\_mutex) locks range.

We use 8 different mutex in this project.

Some example outputs of our project:



