MVP Assignment 3 - Matrix Inversion

Due date: 19/4/2010

1 Analysis

First we will analyse the code from `pmatrics.c` that you discovered in the previous assignment. The example has been updated for this assignment. The purpose is to parallize it using *pthreads*. The code supplied contains helpful comments.

Exercise 1: Analysis of `inv_mat(..)`.

Q: *Identify the different main phases of the matrix inversion algorithm.*

A: ..

Q: *What are the data dependencies between the phases?*

A: ..

2 Preliminaries

Copy your matrix multiplication code from assignments 1 and 2 in the relevant matrix multiplication functions place holders.

Exercise 2: We will need a barrier.

Q: *Implement it.*

```
1
2 /* Your relevant code goes here. NOT the whole file. */
```

Listing 1: Your barrier implementation.

Exercise 3: Your threads will be started in `pthread_inv_mat`. The function lists what needs to be done but it is incomplete.

Q: *Complete the function.*

```
1
2 /* Your relevant code goes here. NOT the whole file. */
```

Listing 2: Function `pthread_inv_mat`.

Your thread will run *job_inv_mat*. As a preliminatry, copy the original matrix inversion code here, skipping its local variable declarations. You will modify it later.

3 Design and Implementation

You should have identified the different phases of the algorithm in the previous question before continuing.

Exercise 4: We consider thread 0 to be the master thread. We can parallelize the initializations (copy and reset) or let the master thread do it. Similarly, we can parallelize the final copy at the end or not. It does not matter much.

Q: *Why is it not important to parallelize the initialization and the final copy?*

A: ..

Exercise 5: The master thread will do the computation that is needed for all threads to continue.

Q: *Use a barrier for synchronization.*

Q: [Optional] *Implement another synchronization where the master thread signals the other thread by a* `sync_post` *on the right* `sync1` *(array of) semaphores and the other threads wait for it.*

Exercise 6: The main computation phase is done in parallel. To avoid load balancing issues, let's use a round-robin distribution of the threads over the rows (the for-loop on i). This can be done simply by letting thread t execute an iteration if $i\%N == t$, N being the total number of threads. It is also possible to rearrange the loop to jump to the right indices directly.

Q: *What is the load balancing issue that we are addressing with the round-robin solution?*

A: ..

Q: *Implement round-robin for the threads.*

Exercise 7: After the main computation phase, the threads need to synchronize before looping again for the master thread to do its job first. In the first synchronization, if you did the optional question, the "other" threads were waiting for the master but here the master needs to wait for all the other threads before proceeding.

Q: *Use a barrier for this second synchronization.*

Q: [Optional] *Implement this synchronization with semaphores (this time on the sync2 semaphore).*

Exercise 8: The back-substitution can be parallelized too. There is a data-dependency between every iteration of the k loop (that you explained in a previous question) so you will use a barrier.

Q: *Parallelize the back-substitution, avoiding load balancing issues using round-robin like previously on the i loop.*

Exercise 9: [Optional] The program outputs different timing results when it executes. You can read the comments in the `timed_call` function to see what it does.

Q: [Optional] *Explain the different outputs of the* `timed_call` *functions.*

A: ..

```
1 /* Your relevant  code goes here.  NOT the whole file. */
```

Listing 3: Matrix inversion with semaphores.

```
1 /* Your relevant  code goes here.  NOT the whole file. */
```

Listing 4: Matrix inversion with barriers.

4 Authors

I/We have solved these exercises independently, and each of us has actively participated in the development of all of the exercise solutions.

Name 1	Name 2
.....	.....
Signature	Signature
Name 3	Name 4
.....	.....
Signature	Signature
Name 5	Name 6
.....	.....
Signature	Signature
Name 7	Name 8
.....	.....
Signature	Signature