MVP Assignment 2

Due date: 24/3/2010

1 Question

Exercise 1: Exercise in the book.

Q: *Exercise 3, chapter 6.*

A: It is important that the operations are atomically, because if they were not atomically, two thread could lock or unlock the Mutex simultaneously and thus run their routines simultaneously.

2 Hello World

Exercise 2: Hello world with threads. Your program

- starts n threads, n is given as a command line argument,
- has every thread print a message of the form `Thread i/n: Hello world!` where `i` is the ID of the thread and `n` is the total number of threads,
- waits for all threads to terminate before terminating.

You are not allowed to use any global variable for that. The point of the exercise is to learn how to prepare data and send them to new threads without race.

Q: *Write a "hello world" program using pthreads.*

```c
typedef struct whatever
{
        int id;
        int total;
} storetype;

void helloworld (void *input)
{
        storetype *t=input;
        int id=(*t).id+1;
        int total =t->total;
        printf ("Thread %d / %d: Hello World! \n",id,total);
        return NULL;
}


/* start_thread and join_threads functions here */

int main (int argc, char* argv[])
{
        int i;
        int total = atoi(argv[1]);
        pthread_t pths[total];
        storetype storage[total];
        for (i=0;i<total;i++)
        {
                storage[i].id=i;
                storage[i].total=total;
                start_thread (&pths[i], helloworld ,&storage[i]);
        }
        join_threads (pths, total);
        exit (EXIT_SUCCESS);
}
```

Listing 1: Hello world with pthreads.

Note: You have a `sample-code.c` file where you will find basic functions to bootstrap you on pthreads.

Exercise 3: [Optional] Additional exercise in the book.

Q: [Optional] *Exercise 1, chapter 6.*

A: ..

3 Mandelbrot's Set

Exercise 4: In this exercise you parallelize a generator of fractals, here Mandelbrot's set. The image is computed as follows: Every point on the picture corresponds to a point in the complex space. We compute the following series of numbers: $z_0 = 0$, $z_{n+1} = z_n^2 + c$ where c is the constant corresponding to the current coordinate. The color n (here modulo 256) is assigned to the pixel whenever $|z| > K$ with K being some constant value. You can read more on wikipedia if you wish.

Q: *Why is this problem a problem trivial to parallelize?*

A: ..

Q: *Parallelize it with pthreads using a 1-D partitioning on the output data.* Your program should use automatically the number of physical processors you have. Run this on a multi-core machine for the experiment to be meaningful. Your partitioning should allocate a block of lines to every processor. You will find some helpful code in `sample-code.c`.

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

Listing 2: Parallelized computation of Mandelbrot's set.

Q: When you benchmark this (with the provided script) you will see some disappointing results. *First, explain the different outputs of the time command, in particular why user time is greater than real time.*

A: ..

Q: The seamingly trivial-to-parallelize program suffers from some very important issue. *What is the problem and where does it come from?* Hint: The next question solves "the problem".

A: ..

Q: *Parallelize it again using a 1-D round-robin partitioning on the output data.* The round robin partitioning assigns each row to a different processor at regular turns. For this exercise, the modification to the previous version should be minor.

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

Listing 3: Parallelized computation of Mandelbrot's set.

4 Parallel Matrix Multiplication

Exercise 5: Your `pmatrix.c` from assignment 1 has been updated in this assignment. First, copy your block matrix multiplication from assignment 1.

Q: *Write the job function for every thread to compute the multiplication by block.* The function is already here and is called `job_block_mat_mult`. Hint: Use the macros DECOMPOSE_BY_ROW and DECOMPOSE_BY_BLOCK to ease the programming. This is an easy method. Keen students may rewrite their functions to avoid if-statements at every loop.

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

Listing 4: Parallelized matrix multiplication by block.

5 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