Homework 3

1. (10 points) Ex 4.10 Which of the following components of program state are shared across threads in a multithreaded process?
a. Register values
b. Heap memory
c. Global variables
d. Stack memory

2. (10 points) Ex 4.11 Can a multithreaded solution using multiple user-level threads achieve better performance on a multiprocessor system than on a single processor system? Explain. Assume the many-to-one multithreading model is used.

3. (10 points) Is it possible to have concurrency but not parallelism? Explain.

4. (10 points) Using Amdahl's Law, calculate the maximum possible speedup for the following applications:

- 95% parallelizable with (a) 8 processing cores and (b) 16 processing cores
- 50% parallelizable with (a) 8 processing cores and (b) 16 processing cores

Amdahl's Law only gives the ideal case, give an example why real systems never achieve the maximum possible speedup.

5. (10 points) Ex 4.16 A system with two dual-core processors has four cores available for scheduling. A CPU-intensive application is running on this system. All input is performed at program start-up, when a single file must be opened. Similarly, all output is performed just before the program terminates, when the program results must be written to a single file. Between startup and termination, the program is entirely CPU bound. Your task is to improve the performance of this application by multithreading it. The application runs on a system that uses the one-to-one threading model (each user thread maps to a kernel thread).

- How many threads will you create to perform the input and output? Explain.
- How many threads will you create for the CPU-intensive portion of the application? Explain.

6. (10 points) Modify the code below so that the following requirements are meet.

The program is required to perform the following **in the order given**.

1. All four threads in the child process run concurrently and run to completion (i.e. they all print "THREAD X FINISHED"). The order of thread completion is not important.
2. The child process runs to completion (i.e. it prints "CHILD PROCESS FINISHED").
3. The parent process runs to completion (i.e. it prints "PARENT PROCESS FINISHED").

Include your code and an example output in your submission document, no separate file required.

```c
#include <pthread.h>
#include <stdio.h>
#include <unistd.h>

int id[4];

/* the thread */
void *runner(void *param) {
    int *id = (int *)param;
    printf("THREAD %d FINISHED\n", *id);
    pthread_exit(0);
}

int main(int argc, char *argv[]) {
    pid_t pid;
    pthread_t tid[4];
    pthread_attr_t attr;

    pid = fork();

    if (pid == 0) {
        /* child process */
        pthread_attr_init(&attr);
        for (int i=0; i<4; i++) {
            id[i] = i;
            pthread_create(&tid[i], &attr, runner, &id[i]);
        }
        printf("CHILD PROCESS FINISHED\n");
    } else if (pid > 0) {
        /* parent process */
        printf("PARENT PROCESS FINISHED\n");
    }

   return 0;
}
```