NATIONAL UNIVERSITY
OF COMPUTER & EMERGING SCIENCES
PESHAWAR CAMPUS

Problem Set:	Assignment 05	**Semester**:	Spring 2013
Points:	10		
Date Set:	See Slate	**Due Date**:	See Slate
Course:	CS206 Operating Systems	**Instructor**:	Nauman

1 Tasks

Note: Code in the following is intentionally left low-quality. Please write the code yourself.

1. You wrote the following code in your last assignment:

```c
/* Includes */
#include <unistd.h>     /* Symbolic Constants */
#include <sys/types.h>  /* Primitive System Data Types */
#include <errno.h>      /* Errors */
#include <stdio.h>      /* Input/Output */
#include <stdlib.h>     /* General Utilities */
#include <pthread.h>    /* POSIX Threads */
#include <string.h>     /* String handling */

#define NUM_RUNS 10000000

/* prototype for thread routine */
void handler ( void *ptr );

int counter; /* shared variable */

int main() {
    int i[2];
    pthread_t thread_a;
    pthread_t thread_b;

    i[0] = 0; /* argument to threads */
    i[1] = 1;

    pthread_create (&thread_a, NULL, (void *) &handler, (void *) &i[0]);
    pthread_create (&thread_b, NULL, (void *) &handler, (void *) &i[1]);

    pthread_join(thread_a, NULL);
    pthread_join(thread_b, NULL);

    printf("------------------------------------------\n");
    printf("Final counter value: %d\n", counter);
    printf("Error:               %d\n", (NUM_RUNS*2-counter));
    exit(0);
}
```

```c
void handler ( void *ptr ) {
   int iter = 0;
   int thread_num;
   thread_num = *((int *) ptr);
   printf("Starting thread: %d \n", thread_num);

   while(iter < NUM_RUNS) {
      counter++;
      iter += 1;
   }

   printf("Thread %d, counter = %d \n", thread_num, counter);
   pthread_exit(0); /* exit thread */
}
```

As you would recall, it calculated the result incorrectly. Your task in this assignment is to add semaphores as locks to fix this problem.

For your ease, here is what you need to use semaphores:

(a) Including the header: `semaphore.h`

(b) Create a mutex (globally): `sem_t mutex;`

(c) Initialize the mutex (once): `sem_init(&mutex, 0, 1);`

(d) Down operation: `sem_wait(&mutex);`

(e) Up operation: `sem_post(&mutex);`

(f) Destroy the mutex (once): `sem_destroy(&mutex);`

You need to figure out where each of these pieces will go. Fix the code using these (and anything else you think is necessary) so that the final counter value is correct. You must include the final output proving your code is calculating the correct result.

2 Submission

Provide screenshots of everything you have done in one PDF file. Other formats will not be accepted and you will get no credit if you provide another format.

Include detailed screenshots of your working and at the end, include concise answers to the questions posed above.