NATIONAL UNIVERSITY
OF COMPUTER & EMERGING SCIENCES
PESHAWAR CAMPUS

Problem Set:	Assignment 03	**Semester**:	Spring 2020
Points:	10		
Date Set:	See Slate	**Due Date**:	See Slate
Course:	CS220 Operating Systems	**Instructor**:	Dr. Nauman

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

1 Threads and Race Conditions

1. The following code deals with basic thread creation.

   (a) Write the following program and observe and *explain* the output.

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

void* thread1() {
        for (int c = 0; c < 10; c++)
                printf("Hello\n");
}

void* thread2() {
        for (int c = 0; c < 10; c++)
                printf("World\n");
}

int main() {
        int status;
        pthread_t tid1,tid2;

        pthread_create( &tid1, NULL, thread1, NULL );
        pthread_create( &tid2, NULL, thread2, NULL );

        pthread_join( tid1, NULL );
        pthread_join( tid2, NULL);

        return 0;
}
```

   (b) Modify the program to create four threads using the same two functions (`thread1` and `thread2`)

   (c) Run both versions and include screenshots of the output.

Note To compile the program, you need to include the following switch in the compilation command: `-lpthread`. This tells gcc to use the registered pthread library.

2. Take a look at the following code:

```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);
}

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 */
}
```

(a) Compile and execute the program.

(b) Answer the following questions:

i. What should be the value of the counter variable at the end?

ii. What is the value you get?

iii. How large is the error and how much does it vary on different runs?

iv. How much *user time* (rougly) does the program take to run on your system?

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.