Homework 4

CS211 - Fall 2024

(You are encouraged to use AI tools but you need to mention the tool you used and submit both your prompt and your solution)

This homework is designed to help you become familiar with synchronization in concurrent programming.

Question 1: The code segment below is to increase a variable count 200000 times by using two threads. Run the code (in the attached file hw4_1.c) 10 times to see what is the final value of count and explain why it is not 200000.

```
#define MAX 100000

// Shared variable
int count = 0;

void* increasing(void* args) {
    for (int i = 0; i < MAX; i++)
        count++;
    return NULL;
}

int main(int argc, char* args[]) {

    pthread_t even_thread, odd_thread;
    pthread_create(&even_thread, NULL, increasing, NULL);
    pthread_create(&odd_thread, NULL, increasing, NULL);

    pthread_join(even_thread, NULL);
    pthread_join(odd_thread, NULL);

    printf("Finally, n = %d\n", count);

    return 0;
}
```

Question 2: The pthread library provides us functions to create and use a lock
https://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread_mutex_init.html
https://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread_mutex_lock.html
https://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread_mutex_unlock.html
https://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread_mutex_destroy.html
You are required to fix the code in question 1 by creating a lock and adding the code of locking/unlocking to ensure the final value of count is always 200000.

Question 3: Use the lock of the pthread library to fix the code in question 2 of HW3 to ensure the sequence is always printed out as 0, 1, …, 2n+1.

Question 4: Instead of using the lock from pthread, we can implement our own lock using atomic operations from the standard library stdatomic.h.
Your task is to implement a spin lock (version 3.0) using only the function atomic_flag_test_and_set
https://pubs.opengroup.org/onlinepubs/9799919799/functions/atomic_flag_test_and_set.html#
Then test your lock with the code in question 1 above.

Question 5: The lock implemented in question 4 uses 'busy-waiting.' To avoid this, we need a system call to make threads sleep or wake up other threads. In Linux and recent versions of macOS, we can use futex
https://en.wikipedia.org/wiki/Futex
And here is the manual on how to use it
https://man7.org/linux/man-pages/man2/futex.2.html
Your task is to implement the lock (version 4.0) as discussed in the lecture using futex, and then test your lock with the code from question 1.
Do you see the new implementation helps the program (in question 1) run faster?