CS575 Homework 4 - James Young

a. Identify the data involved in the race condition.

The data involved in the race condition is `available_resources`.

b. Identify the location (or locations) in the code where the race condition occurs.

The race condition occurs in the locations `decrease_count` and `increase_count`.

In `decrease_count`, if two or more processes call this function at the same time, it may think there are enough resources and decrement `available_resources` to a negative value. Similarly, in `increase_count`, if two or more processes increment `available_resources` at the same time, it may lead to wrong values.

c. Using a semaphore or mutex lock, fix the race condition. It is permissible to modify the decrease count() function so that the calling process is blocked until sufficient resources are available.

Using mutex lock, the mutex is locked before any changes are made to `available_resources` in both the `decrease_count` and `increase_count` functions. This ensures that only one process can change it at a time. After `available_resources` is modified, the mutex is unlocked to allow other processes to access it.

Below shows the code changes:

```
# define MAX_RESOURCES 5
int available_resources = MAX_RESOURCES;
pthread_mutex_t mutex; // ADD - Created a mutex lock

/* decrease available_resources by count resources */
/* return 0 if sufficient resources available, */
/* otherwise return -1 */
int decrease_count(int count) {
    pthread_mutex_lock(&mutex); // ADD - Lock mutex before modifying available resources
    if (available_resources < count) {
        pthread_mutex_unlock(&mutex); // ADD - Unlock mutex since no changes to available re
        return -1;
    } else {
        available_resources -= count;
        pthread_mutex_unlock(&mutex); // ADD - Unlock mutex after modifying available_resou
        return 0;
    }
}
```

```c
/* increase available_resources by count */
int increase_count(int count) {
    pthread_mutex_lock(&mutex); // ADD - Lock mutex before modifying available_resources
    available_resources += count;
    pthread_mutex_unlock(&mutex); // ADD - Unlock mutex after modifying available_resources
    return 0;
}
```