Lab 13

1. Write a C program that creates three threads. The threads will keep adding random numbers between -500 and +500 to a shared variable that initially has the value 0. The threads will terminate when the shared variable has an absolute value greater than 500.

2. Write a C program that receives strings containing any characters as command-line arguments. The program will create a frequency vector for all lowercase letters of the alphabet. The program will create a thread for each command-line argument, each thread will update the letter frequency vector based on the characters present in its corresponding command-line argument. Use efficient synchronization.

3. Write a C program that takes as command-line arguments 2 numbers: N and M. The program will create N threads and simulate a race where each thread must pass through the M checkpoints. Through each checkpoint, the threads must pass one at a time (no 2 threads can be inside the same checkpoint at the same time). Each thread that enters a checkpoint will wait a random amount of time between 100 and 200 milliseconds (usleep(100000) makes a thread or process wait for 100 milliseconds) and will print a message indicating the thread number and the checkpoint number, then it will exit the checkpoint. Ensure that each thread waits until all threads have been created before starting the race.

4. Write a C program that reads a number N and creates 2 threads. One of the threads will generate an even number and will append it to an array that is passed as a parameter to the thread. The other thread will do the same but with odd numbers. Implement synchronization between the two threads so that they alternate in appending numbers to the array until they reach the maximum length N.