Lab 13

1. Write a C program that creates N threads (N given as a command line argument). 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. Ensure proper synchronization Print a message every time a thread modifies the variable.

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 reads a number N from standard input and generates a random number X (between 0 and 1000). The main process creates N threads that attempt to guess the number X in an infinite loop and one other thread that waits until one of the N threads guesses the number X. The first of the N threads that guesses correctly the number X will notify the waiting thread. The waiting thread will print a message indicating which one of the N threads guessed the number. All threads will terminate after the number is correctly guessed.

4. Write a C program that creates N threads (N given as a command line argument). The main process opens a file F, provided as a command line argument (the file's contents are words of a maximum of 20 characters each separated by spaces). Each thread will take turns reading between 1 and 3 words from the file and concatenating them to a thread-local buffer until all the content of the file is read. Once the whole file is completely read, the threads return their local buffer and the main process will print the result from each thread. Ensure that each thread, after it does one reading pass, waits for the other threads to complete their reading attempt before starting a new reading pass.