Question 9: Modify server & explain your modifications.

Answer: The *mutex* is locked in writeLog, just before the buffer is accessed and written to. This ensures that the critical region with the shared data (the buffer logBuffer) is only allowing one thread to access it at a time. If the lock is unlocked, the thread will be able to immediately enter the critical section. If the lock is already locked, the thread trying to access the critical region will be blocked.

As for where in the code the lock of the mutex happens, it is safe for it to do it *right before* the call to *sprintf*, and there's no need to do it earlier, such as before something is written into myBuffer. This is because myBuffer is a local variable and only accessible to the single thread itself. Therefor the mutex does not have to be locked for a thread to write into myBuffer.

When the logging is done, in logToFile, it unlocks the mutex (and logReady is set to 0, to indicate that the log in the buffer was written). If there had been another thread trying to access the mutex while it was locked, this thread (which was previously blocked) is now allowed to access the data and buffer.

```
// Initialize the mutex
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

int main(int ac, char **av)
{
        pthread_t logThread;

        logfptr = fopen("webserver.log", "w");

        if(logfptr == NULL)
        {
                fprintf(stderr, "Cannot open log file\n");
                exit(-1);
        }

        // The thread is created responsible to log safely
        pthread_create(&logThread, NULL, logToFile, NULL);

        startServer(PORTNUM);

        // Waiting for the logging thread to finish
        pthread_join(logThread, NULL);
}
```

```
void writeLog(const char *format, ...)
{
        char myBuffer[LOG_BUFFER_LEN];
        va_list args;

        va_start(args, format);
        vsprintf(myBuffer, format, args);
        va_end(args);

        while(logReady == 1)

        // Thread will try to lock the mutex
        pthread_mutex_lock(&mutex);
        // Thread enters if the lock was unlocked (else it is blocked)

        sprintf(logBuffer, "%s: %s", getCurrentTime(), myBuffer);
        logReady = 1;
}
```

```
void *logToFile(void *n){
    while (1){
        while(logReady == 0)

        fprintf(logfptr,"%s",logBuffer);
        fflush(logfptr);
        logReady = 0;

        // When safely logged, the mutex is unlocked
        pthread_mutex_unlock(&mutex);
    }

    fclose(logfptr);
    pthread_exit(NULL);
}
```