Graded task 3 on Threads mutexes and signals

Create multi-threaded guessing game. The program should take two parameters: **-n –** size of a vector and **-t** number of threads. Main program creates and initializes a vector of size **n** with zeros. The main thread creates **t** worker threads which share the vector. The main thread awaits signals. On SIGINT (C-c) it puts a random positive non zero value [1, 255] to random position in the vector. GUESSED_VALUE is an integer that is shared among all threads initialized with zero. On SIGUSR1 it prints a value GUESSED_VALUE, then cancels a random thread. On SIGQUIT (C-\) it cancels all threads and quits. Moreover, the main thread prints the entire vector to stdout every 500ms.

Each worker thread works in a loop. It chooses random value from the vector. If it is zero, do nothing. If it is equal to GUESSED_VALUE then send SIGUSR1 to current process. If it is any other positive non zero value then set GUESSED_VALUE equal to it. Then the thread sleeps for 1 second.

Vector operations and operations on GUESSED_VALUE should be synchronized with two different mutexes. Mutexes should be locked and unlocked properly considering the thread cancel functionality.

Graded stages (please report the stage as soon as you finish it).

1. Main program reads input parameters, creates and initializes the vector, creates **t** threads and waits for them to finish. Threads start their work function, print their unique thread id to stdout and finish immediately.
2. Add entire thread loop functionality without sending SIGUSR1. Access to vector and GUESSED_VALUE should be synchronized.
3. Add SIGINT (C-c) signal handling and printing vector to stdout every 500ms.
4. Add SIGUSR1 signal handling and sending, on main thread and worker threads.
5. Add SIGQUIT (C-\) signal handling