Lab 2 – Answers:

Q1:

Guard class– Guard is a simple design pattern made for prevention of 2 threads

Sharing the same scope. By doing that, we ensure each thread will do its job without interruptions.

When the thread finishes up its duty, the Guard will unlock it and will move on to the next

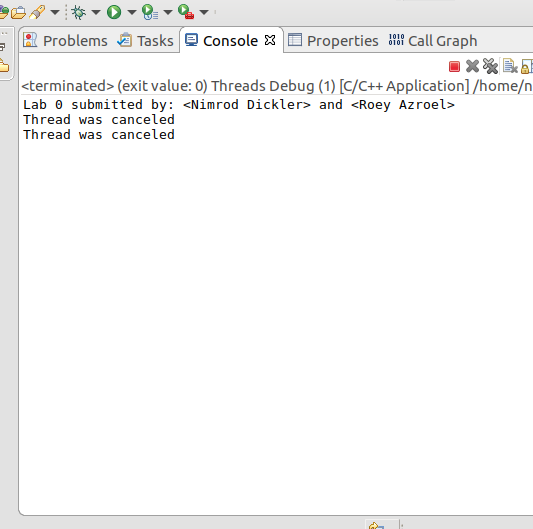
Thread.

MThread class – Mthread is the class that activates the treads.

It has the start() function that creates the thread, the waitForThread()

Function that adds the tread to the scope and a destructor that kills threads if needed.

Q2:



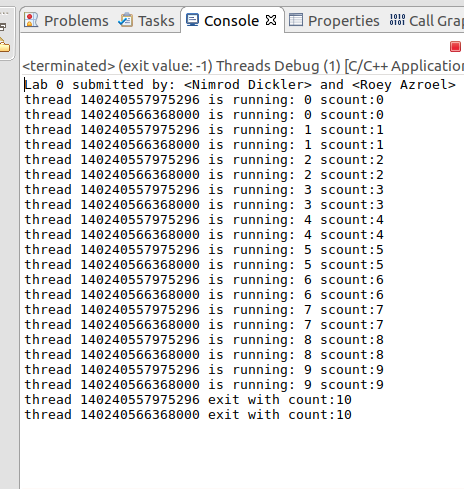
The program creates two variables mt1 and mt2 of type MyThread.

It then activates them using the start() function, which creates a new tread,

And runs it.

In our case, after the threads started, the main tread reaches the delete commands first, so the MThread’s destructor is activated, Killing the threads, and printing “Thread was canceled” one at a time.

Q3:



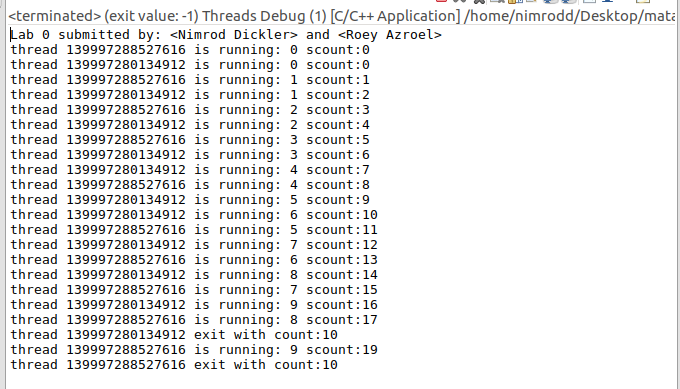
The outcome of the program has changed. When the threads are created, the

pthread \_join() function is activated afterwards, making the main tread to wait until the other threads finish their roll. This makes the program to output the counting, without interruption.

While running, they each gain access to the variable scount, and each, in turn, increments it.

When they are finished, the main tread activates the destructor, but not printing because the tread’s id had already changed to -1.

Q4:



Much like the answer above, the treads are activated without interruptions from the main tread.

The difference is that in this section, we use the Guard class.

The Guard class uses the mutex variable, which allows it to lock a tread from using the same resource

As the other tread, hence letting the scount variable to be different on every count as shown above.