**Lab 1 – Threads**

1. Explanation:  
   MTHREAD – is the class which implementing the thread behavior, it is calling in another thread the “run()” function which counting in a while loop to 10 and then stop and finish the work.  
     
   Guard – the function responsible for the mutex implementation, it locks the object to prevent from the scount object to be over written by cross threads that use the same Counter class.
2. Print of output:  
   **Lab 1 submitted by: Saar Sinai and Kobi Mizrahi  
   thread 140041206241024 is running: 0 scount:0  
   Thread was canceled**

**Thread was canceled**

The operation starting 2 thread and then deleting them. In between one thread was fast enough to print once that it is running and then we received the cancelation messages from both threads.

1. Print of output

**Lab 1 submitted by: Saar Sinai and Kobi Mizrahi**

**thread 139926173878016 is running: 0 scount:0**

**thread 139926182270720 is running: 0 scount:0**

**thread 139926173878016 is running: 1 scount:1**

**thread 139926182270720 is running: 1 scount:1**

**thread 139926173878016 is running: 2 scount:2**

**thread 139926182270720 is running: 2 scount:2**

**thread 139926173878016 is running: 3 scount:3**

**thread 139926182270720 is running: 3 scount:3**

**thread 139926182270720 is running: 4 scount:4**

**thread 139926173878016 is running: 4 scount:4**

**thread 139926173878016 is running: 5 scount:5**

**thread 139926182270720 is running: 5 scount:5**

**thread 139926182270720 is running: 6 scount:6**

**thread 139926173878016 is running: 6 scount:6**

**thread 139926182270720 is running: 7 scount:7**

**thread 139926173878016 is running: 7 scount:7**

**thread 139926182270720 is running: 8 scount:8**

**thread 139926173878016 is running: 8 scount:8**

**thread 139926182270720 is running: 9 scount:9**

**thread 139926173878016 is running: 9 scount:9**

**thread 139926182270720 exit with count:10**

**thread 139926173878016 exit with count:10**

since we used for both threads the function “waitForThread” the main function did what we told it and waited for it to announce that it finished executing the work. So we now see that both of them printed 10 times the message that was expected.

1. Print output:

**Lab 1 submitted by: Saar Sinai and Kobi Mizrahi**

**thread 140251197523712 is running: 0 scount:0**

**thread 140251205916416 is running: 0 scount:0**

**thread 140251197523712 is running: 1 scount:1**

**thread 140251205916416 is running: 1 scount:2**

**thread 140251197523712 is running: 2 scount:3**

**thread 140251205916416 is running: 2 scount:4**

**thread 140251197523712 is running: 3 scount:5**

**thread 140251205916416 is running: 3 scount:6**

**thread 140251197523712 is running: 4 scount:7**

**thread 140251205916416 is running: 4 scount:8**

**thread 140251197523712 is running: 5 scount:9**

**thread 140251205916416 is running: 5 scount:10**

**thread 140251197523712 is running: 6 scount:11**

**thread 140251205916416 is running: 6 scount:12**

**thread 140251197523712 is running: 7 scount:13**

**thread 140251205916416 is running: 7 scount:14**

**thread 140251197523712 is running: 8 scount:15**

**thread 140251205916416 is running: 8 scount:16**

**thread 140251197523712 is running: 9 scount:17**

**thread 140251205916416 is running: 9 scount:18**

**thread 140251197523712 exit with count:10**

**thread 140251205916416 exit with count:10**

now we added the Guard function which uses the Mutex object to stop the option to continue a thread if another thread already locked the mutex object, that is for other thread not override data in the same function (as happened in the last run) and now we see the scound object not over run the same values for both the thread.