**Thread Management using POSIX Library - Detachment and Cancellation assignments**

Mandatory

1. Write a program to create 3 threads with the detach and cancel status as below.

Thread# Detached (Y/N) Cancel type Cancel status

1 Y PTHREAD\_CANCEL\_DISABLE

2 N PTHREAD\_CANCEL\_ENABLE PTHREAD\_CANCEL\_DEFERRED

3 N PTHREAD\_CANCEL\_ENABLE PTHREAD\_CANCEL\_ASYNCHRONOUS

a. Let all the threads read and display their detach, cancel type and status and then display thread specific message as below.

T1: Display message in the format as below every 2 secs

<timestamp> Health OK

T2: Print numbers starting from 1000 in steps of 2 at an interval of 3 secs in format as below.

<timestamp> <threadid> <countvalue>

T3: Print numbers starting from 2000 in steps of 2 at an interval of 3 secs

<timestamp> <threadid> < countvalue >

b. After creating threads, and after 3 minutes from main(), cancel all 3 threads

c. From an other terminal, use command below to view the thread count of your program

§ ps -eLF

§ top [For top command usage to refer https://www.golinuxcloud.com/check-threads-per-process-count-processes/ ]

d. What difference did you observe between top and ps command?

Ans:

* The ps command provides a snapshot of the current processes and their threads at the moment it is run.
* The top command provides a dynamic, real-time view of the processes and their threads, updating continuously.

e. Which column shows the number of threads in ps and in top commands?

Ans:

* In the ps command, the number of threads is shown in the NLWP (Number of Light Weight Processes) column.
* In the top command, the number of threads is shown in the TH (Threads) column.

f. Check the last message timestamp from the threads

Ans: The last message timestamp by looking at the output of the program before the threads are cancelled. The timestamps will be printed in the format provided in the program.

g. Which thread was cancelled first and why?

Ans: Thread 3 (T3) was likely cancelled first because it uses PTHREAD\_CANCEL\_ASYNCHRONOUS, which means it can be cancelled immediately upon receiving the cancel request

h. Were all 3 threads cancelled? Justify the observation

Ans: Yes, all three threads were cancelled. Thread 1 (T1) was cancelled after the main thread called pthread\_cancel, even though it had PTHREAD\_CANCEL\_DISABLE initially, it would be cancelled once the main thread exits. Thread 2 (T2) and Thread 3 (T3) were cancelled as per their cancel types, with T3 being cancelled immediately and T2 being cancelled at the next cancellation point.

