CS 307: Systems Practicum
Activity 1: Synchronization and Threading
To be done in class groups; due: before the end of the class on 18th Jan 2024

Q1. Implement the example of MyThread class in Python using the threading module with two threads thread 1 and thread 2 started one after the other (similar to the one shown in the class). [1]

Q2. In the implementation in Q1, check the effect of the delay variable and the effect of the counter variable passed to the print_time function. [1]

Q3. Create a daemon thread and two non-daemon threads. Now, start the two non-daemon threads of delay 6 and 3 seconds, respectively. Now, start a daemon thread that prints the time. Show the running of the daemon and non-daemon threads. [1]

Optional Bonus: +1 points
Q4. Consider the following Python program:

```
import threading
x = 0

def increment_global():
  global x
  x += 1

def taskofThread():
  for _ in range(50000):
    increment_global()

def main():
  global x
  x = 0
  t1 = threading.Thread(target= taskofThread)
  t2 = threading.Thread(target= taskofThread)
  t3 = threading.Thread(target= taskofThread)
  t1.start()
  t2.start()
  t3.start()
  t1.join()
  t2.join()
  t3.join()

if __name__ == "__main__":
```

```
  for i in range(5):
      main()
      print("x = {1} after Iteration {0}".format(i,x))
```

Does this program suffer from a race condition? If yes, then why yes? Please remove the race condition using locks.