Short Assignment - Concurrency

Assume we have a Python-like language which provides mutex and condition variable classes with the following signatures:

```
class mutex:
  method lock()
  method unlock()

class condition:
  method wait(mutex m)
  method signal()
```

We define two additional classes, pmutex and rendezvous, shown here with numbered lines:

```
   class pmutex:
      mutex m
      method lock():
1        print('locking...')
2        m.lock()
3        print('locked...')

      method unlock():
4        print('unlocking:')
5        m.unlock()

   class rendezvous:
      pmutex P
      condition C
      count = 0

      method meetup():
6        P.lock()
7        count = count+1
8        while count < 2:
9           C.wait(P.m)
10       C.signal()
11       P.unlock()
```

Note – consider line 8 (“while count < 2”) to execute each time the condition is tested, followed by line 9 (if the condition is true) or 10 (if false).
Finally we have the following code, again with line numbers:

```
     rendezvous R
     function thread_A():
12      R.meetup()

     function thread_B():
13      sleep(1 second)
14      R.meetup()

     function thread_C():
15      sleep(1 second)
16      R.meetup()
```

Answer format: Your answer will specify one or more **execution sequences**, consisting of a sequence of thread ID / line number pairs – e.g. an execution sequence might start with A/12,A/6, etc. (ignore output from “print” commands) An execution sequence ends when all threads have either returned from their thread function, or are waiting forever on a condition.

Deliberable: Assuming all three threads begin execution at the same time, give ****two**** (2) different legal execution sequences.

Submission instructions: Please submit your answer via Canvas, either in PDF format or via the text entry field.