SUBMISSION GUIDELINES:

For each question, please modify the corresponding qXX.py file to contain the required answers and solutions.

When a textual answer is called for, write your answer as a comment.

When a coding answer is called for, modify the skeletal code we have provided to solve the problem, making sure your solution makes progress whenever it is feasible to do so, and obeys the safety criteria described in the problem description.

Your code must be able to execute correctly when invoked, as that is how it will be tested.

IMPORTANT:

All of your answers should follow the commandments, located here: http://www.cs.cornell.edu/Courses/cs4410/2015fa/papers/commandments.pdf

Submit your answers on CMS.

FAQ

1. Are we allowed to take advantage of the optional argument to the acquire method that causes the semaphores to be non-blocking and return true or false if the acquire does not succeed?

   No, for MP1, please stick with the semaphore interface discussed in class (init, P, V). It's clear and simple to reason about.

2. How can I use monitors in Python? There is no object called monitor; only Locks, Semaphores and Condition variables. Should I create a Monitor class?

   In Python the behaviors of Monitors are realized by Condition() objects that carry a Lock() inside them. By acquiring and releasing this lock you enter the monitor and then you can wait, notify and notifyAll on the Condition Object(s).

   Here is a pseudo example:

```
class PrinterRoom():

     def __init__(self):
        self.printerBusy = False
        self.printerLock = Lock()
        self.printerAvailable = Condition(self.printerLock)

     def waitForPrinter(self):
        with self.printerLock:          #acquire the monitor lock
            while self.printerBusy:    #check if you need to wait
                self.printerAvailable.wait()  # wait until notify()
            self.printerBusy = True    #set printerBusy to True,
                                        #to make others wait

     def doneWithPrinter(self):
        with self.printerLock:          #acquire the monitor lock
            self.printerBusy = False   #set to False when done
            self.printerAvailable.notify() #notify 1 person waiting
                                        #for printer to be available

class Student():

    def printDocument(self):
        pr = PrinterRoom()
        while True:
            pr.waitForPrinter()
            #use printer now
            pr.doneWithPrinter()
```

For more information on synchronization objects in Python, you can read the Python threading module documentation.
(https://docs.python.org/2/library/threading.html)