Name:
ID #:

Quiz 2
CMPS 405: Operating Systems
Fall 2015
Dr. Ryan Riley

Consider the following description of the cigarette smoker's problem:

> Assume a cigarette requires three ingredients to make and smoke: tobacco, paper, and matches. There are three smokers around a table, each of whom has an infinite supply of one of the three ingredients — one smoker has an infinite supply of tobacco, another has paper, and the third has matches.
>
> There is also a non-smoking agent who enables the smokers to make their cigarettes by randomly selecting two of the supplies to place on the table. The smoker who has the third supply should remove the two items from the table and use them (along with their own supply) to make a cigarette, which they smoke for a while. Once the smoker has finished his cigarette, the agent places two new random items on the table. This process continues forever.

Consider the following pseudo-code for a semaphore based solution to the problem:

```
init() {
        agent = Semaphore(1)
        paper = Semaphore(0)
        matches = Semaphore(0)
        tobacco = Semaphore(0)
}
```

```
agent() {
  while(1) {
        agent.acquire()
        r = rand(0,3)
        if (r == 0) {
          paper.release()
          matches.release()
        }
        else if (r == 1) {
          matches.release()
          tobacco.release()
        }
        else if (r == 2) {
          paper.release()
          tobacco.release()
        }
  }
}
```

```
tobacco_smoker() {
  while(1) {
    paper.acquire()
    matches.acquire()
    smoke()
    agent.release()
  }
}
```

```
paper_smoker() {
  while(1) {
    matches.acquire()
    tobacco.acquire()
    smoke()
    agent.release()
  }
}
```

```
matches_smoker() {
  while(1) {
    paper.acquire()
    tobacco.acquire()
    smoke()
    agent.release()
  }
}
```

a. This solution may cause deadlock. Describe the details of a scenario where deadlock would occur.

There are many possible scenarios. The important thing is that a specific scenario is given that demonstrates deadlock. Here is one:

1. Assume tobacco_smoker runs first and then calls paper.acquire(). It will go to sleep because that semaphore is currently 0.
2. Next, paper_smoker runs and calls matches.acquire(). It will go to sleep. It will go to sleep because that semaphore is currently 0.
3. Next, matches_smoker runs and calls paper.acquite(). It will go to sleep. It will go to sleep because that semaphore is currently 0.
4. Next, the Agent runs, decrements the agent semaphore, generates the random number 0, signals the paper and matches semaphores, then the loop restarts and it calls agent.acquire() again. It will go to sleep because that semaphore is currently 0.
5. Paper_smoker now wakes up (because matches got signaled), decrements the matches semaphore, and calls tobacco.acquire(). It will go to sleep because that semaphore is currently 0.
6. Tobacco_smoker now wakes up (because paper got signaled), decrements the paper semaphore, and calls matches.acquire(). It will go to sleep because that semaphore is currently 0.
7. Now, all threads are asleep and none will ever wakeup.

b. Imagine another solution that doesn't cause deadlock uses the following code for the tobacco smoker:

```
tobacco_smoker() {
   while(1) {
            tobacco_smoker.acquire()
            mutex.acquire()
            // Pick up match
            // Pick up paper
            // Smoke
            mutex.release()
            agent.release()
   }
}
```

Finish this solution by writing the code for init(), agent(), paper_smoker(), and matches_smoker().

Here is an entire solution:

```
init() {
      agent = Semaphore(1)
      lock = Semaphore(1)
      tobacco_smoker = Semaphore(0)
      paper_smoker = Semaphore(0)
      matches_smoker = Semaphore(0)
}

agent() {
    while(1) {
      agent.acquire()
      mutex.acquire()
      r = rand(0,3)
      if (r == 0) {
          tobacco_smoker.release()
      }
      else if (r == 1) {
          paper_smoker.release()
      }
      else if (r == 2) {
          matches_smoker.release()
      }
      mutex.release()
    }
}
```

```
tobacco_smoker() {
    while(1) {
      tobacco_smoker.acquire()
      mutex.acquire()
      // Pick up match
      // Pick up paper
      // Smoke
      agent.release()
      mutex.release()
    }
}

paper_smoker() {
    while(1) {
      paper_smoker.acquire()
      mutex.acquire()
      // Pick up match
      // Pick up paper
      // Smoke
      agent.release()
      mutex.release()
    }
}

matches_smoker() {
    while(1) {
      matches_smoker.acquire()
      mutex.acquire()
      // Pick up match
      // Pick up paper
      // Smoke
      agent.release()
      mutex.release()
    }
}
```