Quiz 4

Questions relating to the Threading API covered in this chapter.

Question # 1

Consider the program below:

```
mutex1 = Mutex.new
mutex2 = Mutex.new

cv = ConditionVariable.new

thread = Thread.new do
  mutex2.synchronize {
    mutex1.synchronize {
      cv.wait(mutex2)
    }
  }
  puts "child thread exiting"
end

# wait for child thread to wait on condition variable
sleep(2)

mutex2.synchronize do
  cv.signal()
end

thread.join()

puts "Main thread exiting"
```

Q What is the outcome of running the above program?

COMPLETED 0%

1 of 1

```
mutex1 = Mutex.new
mutex2 = Mutex.new

cv = ConditionVariable.new

thread = Thread.new do
  mutex2.synchronize {
    mutex1.synchronize {
      cv.wait(mutex2)
    }
  }
  puts "child thread exiting"
end

# wait for child thread to wait on condition variable
sleep(2)

mutex2.synchronize do
  cv.signal()
end

thread.join()

puts "Main thread exiting"
```

Question # 2

In the previous program a junior developer changes the code for the child thread as follows:

```
thread = Thread.new do
  mutex2.synchronize {
    mutex1.synchronize {
      cv.wait(mutex1)
    }
  }
  puts "child thread exiting"
end
```

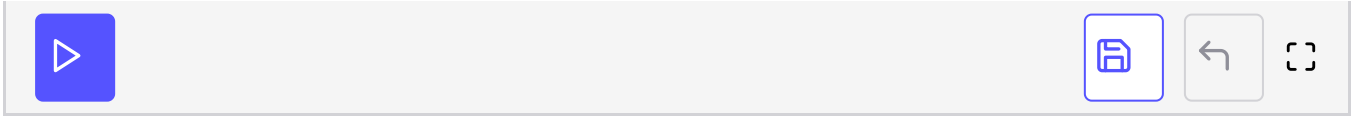
Q What is the outcome of the program now?

COMPLETED 0%

1 of 1

```
mutex1 = Mutex.new
mutex2 = Mutex.new

cv = ConditionVariable.new

thread = Thread.new do
```

```
  mutex2.synchronize {
    mutex1.synchronize {
      cv.wait(mutex1)

    }
  }
  puts "child thread exiting"
end

# wait for child thread to wait on condition variable
sleep(2)

mutex2.synchronize do
  cv.signal()
end

thread.join()

puts "Main thread exiting"
```



Question # 3

Consider the snippet below:

```
monitor = Monitor.new
cv = monitor.new_cond()

Thread.new do
  monitor.mon_synchronize do
    cv.wait()
  end
end

# wait for child thread to wait on condition variable
sleep(1)

cv.signal()
```

Q

If you receive the above snippet for code review, what feedback would you give?

COMPLETED 0%

1 of 1

Question # 4

John knows that standard Ruby (MRI) has a global interpreter lock and sees the following snippet of code:

```
counter += 1
```

Q

John reasons that the snippet is safe as there can only be one thread executing the statement at any time because of GIL. Which of the following is true?

Question # 5

Kim knows that a context switch for a thread doesn't happen in the midst of executing a C method.

Q

Can she avoid using thread synchronization if she’s certain a particular Ruby statements translates to a single C method execution?

Check Answers