Quiz 1

Questions relating to the Threading API are covered in this lesson.

Question#1

Consider the snippet below:

```
Thread.current.name = "mainThread"

# Spawn a child thread
thread = Thread.new do

  thread.exit()
  puts("Child thread exiting")
end


thread.join()

puts("Main thread exiting")
```

Q Is the statement “Child thread exiting” get printed?

COMPLETED 0%

1 of 1

```
Thread.current.name = "mainThread"

# Spawn a child thread
thread = Thread.new do

  thread.exit()
  puts("Child thread exiting")
end


thread.join()

puts("Main thread exiting")
```

Question#2

Consider the code snippet below:

```
Thread.new do

  while true
    sleep(1)
  end
  puts("Child thread exiting")
end

puts("Main thread exiting")
```

Q

Is the statement in the spawned thread printed?

COMPLETED 0%

1 of 1

```
Thread.new do

  while true
    sleep(1)
  end
  puts("Child thread exiting")
end

puts("Main thread exiting")
```

Explanation

It might surprise you that the correct answer is maybe even though if you run the widget several times, only the line from the main thread is printed on the console. Threads scheduling can be non-deterministic and it is possible that the spawned thread is immediately scheduled when created and prints on the console, before the main thread exits.

Question#3

Consider the snippet below:

```
"Main thread executing"
Thread.stop
```

Q What is outcome of running the above snippet?

COMPLETED 0%

1 of 1

```
"Main thread executing"
Thread.stop
```

Question#4

Consider the snippet below:

```ruby
mutex = Mutex.new

Thread.new do
  mutex.lock()
  puts("Child thread exiting")
end

# wait for child thread to exit
sleep(2)
puts("Is mutex locked #{mutex.locked?}")
mutex.lock()
```

Q What is the outcome of running the program?

COMPLETED 0%

1 of 1

```
mutex = Mutex.new

Thread.new do
  mutex.lock()
  puts("Child thread exiting")
end

# wait for child thread to exit
sleep(2)
puts("Is mutex locked #{mutex.locked?}")
mutex.lock()
```

Question#5

Consider the snippet below, where the main thread `join()`-s a child thread after the child thread has finished:

```
thread = Thread.new do
  puts "Child thread exits"
end

sleep(2)

# main thread joins an already completed thread
thread.join()
```

Q What is the outcome of the program?

COMPLETED 0%

1 of 1

```
thread = Thread.new do
  puts "Child thread exits"
end

sleep(2)

# main thread joins an already completed thread
thread.join()
```