ECE 421 Assignment 3 Part 2

Winter2019_Group4:

Nathan Klapstein (1449872)

Tony Qian (1396109)

Thomas Lorincz (1461567)

Zach Drever (1446384)

```ruby
require_relative 'helper'

class TortTest < Test::Unit::TestCase
  def setup
    # Do nothing
  end

  def teardown
    # Do nothing
  end

  ######################
  # Thread sort tests
  ######################

  # Pre:
  #   * input array is orderable
  #   * duration is not specified, or either a numerical, or nil
  #   * worker count is not specified, or is a integer greater than or equal
to 0
  # Post:
  #   * output array is a sorted version of the input array
  def test_thread_sort_rand
    unsorted_array = Array.new(10) { rand(-10_000_000...10_000_000) }
    assert_equal(unsorted_array.sort, Tort.thread_sort(unsorted_array, 1))
  end

  # Pre:
  #   * duration is too short for sorting the array
  # Post:
  #   * Timeout exception is raised noting that the sorting failed to execute
within
  #     the time limit specified by duration
  def test_thread_sort_timeout
    unsorted_array = Array.new(100_000_000) { rand(-10_000_000...10_000_000) }
    assert_raise(Timeout::Error) { Tort.thread_sort(unsorted_array, 0.1) }
  end

  # Pre:
  #   * input enum is orderable
  #   * duration is not specified, or either numerical, or nil
  #   * worker count is not specified, or is a integer greater than or equal
to 0
  # Post:
  #   * output array is a sorted version of the input enum
  def test_thread_sort_enum
    unsorted_enum = Array.new(10) { rand(-10_000_000...10_000_000) }.to_enum
```

```
    assert_equal(unsorted_enum.sort, Tort.thread_sort(unsorted_enum, 1))
  end

  # Pre:
  #   * input array is not orderable
  # Post:
  #   * ArgumentError exception is raised noting non-orderable array
  def test_thread_sort_unsortable
    assert_raise(ArgumentError) { Tort.thread_sort([1, 'foo'], 10) }
    assert_raise(ArgumentError) { Tort.thread_sort('foo', 10) }
  end

  # Pre:
  #   * worker count is a not a integer greater than or equal to 0
  # Post:
  #   * ArgumentError exception is raised noting invalid worker count number
  def test_thread_sort_invalid_workers
    assert_raise(ArgumentError) { Tort.thread_sort([], 10, -1) }
  end

  ######################
  # Process sort tests
  ######################

  # Pre:
  #   * input array is orderable
  #   * duration is not specified, or either numerical, or nil
  #   * worker count is not specified, or is a integer greater than or equal
to 0
  # Post:
  #   * output array is a sorted version of the input array
  def test_process_sort_rand
    unsorted_array = Array.new(10) { rand(-10_000_000...10_000_000) }
    assert_equal(unsorted_array.sort, Tort.process_sort(unsorted_array, 1))
  end

  # Pre:
  #   * duration is too short for sorting the array
  # Post:
  #   * Timeout exception is raised noting that the sorting failed to execute
within
  #     the time limit specified by duration
  def test_process_sort_timeout
    unsorted_array = Array.new(100_000_000) { rand(-10_000_000...10_000_000) }
    assert_raise(Timeout::Error) { Tort.process_sort(unsorted_array, 0.1) }
  end

  # Pre:
```

```
  #   * input enum is orderable
  #   * duration is not specified, or either numerical, or nil
  #   * worker count is not specified, or is a integer greater than or equal
to 0
  # Post:
  #   * output array is a sorted version of the input enum
  def test_process_sort_enum
    unsorted_enum = Array.new(10) { rand(-10_000_000..10_000_000) }.to_enum
    assert_equal(unsorted_enum.sort, Tort.process_sort(unsorted_enum, 1))
  end

  # Pre:
  #   * input array is not orderable
  # Post:
  #   * ArgumentError exception is raised noting non-orderable array
  def test_process_sort_unsortable
    assert_raise(ArgumentError) { Tort.process_sort([1, 'foo'], 10) }
    assert_raise(ArgumentError) { Tort.process_sort('foo', 10) }
  end

  # Pre:
  #   * worker count is a not a integer greater than or equal to 0
  # Post:
  #   * ArgumentError exception is raised noting invalid worker count number
  def test_process_sort_invalid_workers
    assert_raise(ArgumentError) { Tort.process_sort([], 10, -1) }
  end
end
```