CSCI 356 – Programming Assignment 2

Output from tests (see last file for test running script):

```
~/Dropbox/Documents/Terms/2017-09 - Fall/CSCI 356/p-assig2 rake
Tests for main.rb

1. Test with matching.tape
        jkl{abba((yeyey)<<><riroio>>)}asdf{[a],[pp]}vbvbv
ACCEPT

    Machine halted with status true.

2. Test with only_ascii.tape
        asdfpoiu
ACCEPT

    Machine halted with status true.

3. Test with leading_right0.tape
        sdf]{asdf((yeyey)<<><riroio>>)}asdf{[a],[pp]}vbvbv
REJECT
    Mistake while tape at position 4.
    Tape head currently reads "]".
    Expected to see match for nil.
    Halted in state q0.

    Machine halted with status false.

4. Test with missing_left.tape
        yey{e}y)<>
REJECT
    Mistake while tape at position 8.
    Tape head currently reads ")".
    Expected to see match for nil.
    Halted in state q1.

    Machine halted with status false.
```

```
5. Test with mismatch.tape
        yey{ey)<>
REJECT
    Mistake while tape at position 7.
    Tape head currently reads ")".
    Expected to see match for "{".
    Halted in state q1.

    Machine halted with status false.

6. Test with missing_right.tape
        sdf{asdf((yeyey)<<><riroio>>)asdf{[a],[pp]}vbvbv
REJECT
    Mistake while tape at position 49.
    Tape head currently reads "\n".
    Expected to see match for "{".
    Halted in state q1.

    Machine halted with status false.

 ~/Dropbox/Documents/Terms/2017-09 - Fall/CSCI 356/p-assig2
```

Source Code

```
#! /usr/bin/env ruby
#
# file: main.rb

require './pda-oop'

m = PDA.new
m.feed_tape STDIN
status = m.execute_transitions

puts "\n    Machine halted with status #{status}."
exit 0
```

```
# encoding: UTF-8
#
# file: delimiter_matcher.rb

require './tape'
```

```
# This implementation of a deterministic PDA specifically matches delimiters.
# Feed it a tape before executing transitions.
#
# Deterministic Pushdown Automata
# Formally defined as a septuple:
#   M = (𝑸, 𝚺, 𝚪, 𝜹, q₀, Z₀, F)
# This implementation is constructed as:
#   m = DelimiterMatcher.new
#
class DelimiterMatcher

  def initialize
    @stack_alphabet = ['(', '<', '[', '{']
    @start_state = :q0

    # to be matched by index
    @match_alphabet = [')', '>', ']', '}']

    # Actual PDA memory
    @stack = []
    @state = ''
    @tape
  end

  # The same pda object can be used on different input tapes.
  # It will always reset the state and stack.
  def feed_tape(input)
    @tape = Tape.new input
    @state = @start_state
    @stack = []
  end

  def execute_transitions
    send @start_state
  end

  private

  # States transitions:
  #   1. end of tape and stack  (accepting)
  #   2. left delim             y, ⊥/y
  #   3. ascii (not right)      x, ⊥/λ
  #   4. not ascii              (crashing)
  #  (5.) right delim?          (crashing)
  #
  def q0
    @tape.shift_head

    if @tape.end? && @stack.empty?
      puts 'ACCEPT'
      return true
    end
```

```
  # y, ⊥/y
  if @stack_alphabet.include? @tape.head
    # stack was empty
    @stack.push @tape.head
    return q1
  end

  # x, ⊥/λ
  if char_is_ascii?
    return q0
  end

  crash
end

# State transitions:
#   1. end of tape & stack  λ, ⊥/λ
#   2. left delim           y, z/yz     where z is some delim, not empty
#   3. right delim          y', y/λ
#   4. ascii                x, y/y
#   5. not ascii            (crashing)
#   6. right with wrong left(crashing)
#  (7.) right with no left  (crashing)
#   8. end of tape          (crashing)
def q1
  @state = :q1
  @tape.shift_head

  # y, z/yz       where z is some delim, not empty
  if @stack_alphabet.include? @tape.head
    @stack.push @tape.head
    return q1
  end

  # y', y/λ
  if tape_head_matches_stack?
    @stack.pop
    return q1
  end

  # λ, ⊥/λ
  if @tape.end? && @stack.empty?
    return q0
  end

  # x, y/y
  if char_is_ascii?
    return q1
  end
```

```
    crash
  end

  # halt: accept, crash or reject
  def crash
    puts "REJECT"
    puts "    Mistake while tape at position #{@tape.position}."
    puts "    Tape head currently reads #{@tape.head.inspect}."
    puts "    Expected to see match for #{top_of_stack.inspect}."
    puts "    Halted in state #{@state.to_s}."
    return false
  end

  # Char on the tape head
  def char_is_ascii?
    return false if @tape.end?
    # Right matching symbols, y', is part of the input alphabet however we
    # cannot consider it valid criteria for transition.
    return false if @match_alphabet.include? @tape.head
    @tape.head.ascii_only?
  end

  def tape_head_matches_stack?
    index = @stack_alphabet.find_index top_of_stack
    return false if index.nil?
    match_from_stack = @match_alphabet[index]
    return true if @tape.head == match_from_stack
    return false
  end

  def top_of_stack
    @stack.last
  end
end
```

```
# file: tape.rb

require 'io/console'

# Responsible for handling the input stream that acts as a tape feed for a PDA.
class Tape
  attr_reader :position

  def initialize(source)
    @source = source
    @head = ''
    # start at position 1, like column 1 in a file
    @position = 0
  end

  # Recognizes '\n' preceeding nil as end.
```

```
  # Not same as checking for empty string (lambda character)
  def end?
    @source.eof?
  end

  # Reads the character currently under the tape head
  def head
    return '' if @head.nil?
    @head
  end

  # Always shifts right
  def shift_head
    # if end of tape (file, stdin), set blank
    @head = get_char
    @position += 1
  end

  private

  def get_char
    @source.getc
  end
end
```

```
# file: Rakefile
#
# This file is used to run test tapes

task default: :test_main

input_files = ['matching',
               'only_ascii',
               'leading_right0',
               'missing_left',
               'mismatch',
               'missing_right']

test_infiles = Proc.new do |script|
  puts "Tests for #{script}.rb"

  input_files.each_with_index do |file, i|
    print "\n#{i+1}. Test with #{file}.tape \n\t"
    system("cat tapes/#{file}.tape")
    system("./#{script}.rb < tapes/#{file}.tape")
  end
  puts ''
end

task :test_main { test_infiles.call 'main' }
task :test_machine { test_infiles.call 'machine' }
```