```ruby
require_relative '../lib/crackle_pop'
puts CracklePop.new.to_s
```

```ruby
# A sample Guardfile
# More info at https://github.com/guard/guard#readme

guard :minitest do
  # with Minitest::Unit
  watch(%r{^test/(.*)\/?test_(.*)\.rb$})
  watch(%r{^lib/(.*/)?([^/]+)\.rb$})     { |m| "test/#{m[1]}test_#{m[2]}.rb" }
  watch(%r{^test/test_helper\.rb$})      { 'test' }

  # with Minitest::Spec
  watch(%r{^spec/(.*)_spec\.rb$})
  watch(%r{^lib/(.+)\.rb$})         { |m| "spec/#{m[1]}_spec.rb" }
  watch(%r{^spec/spec_helper\.rb$}) { 'spec' }

end
```

```ruby
class CracklePop
  def initialize(rule = { 3 => 'Crackle', 5 => 'Pop' }, rules: rule)
    @rules = rules
  end

  # to_s takes an optional argument of a range otherwise simply plays the game
  # from 1 to 100
  def to_s(range = (1..100))
    range.map { |n| what_is(n) }.join("\n")
  end

  def what_is(integer)
    result = @rules.each_key.map do |k|
      (integer % k).zero? ?  @rules[k] : ''
    end.join
    result.empty? && integer || result
  end
end
```

```ruby
require 'minitest/autorun'
require_relative 'spec_helper'
require 'crackle_pop'

describe 'CracklePop' do
  before :each do
    @cp = CracklePop.new
  end

  it 'must return "Crackle" for numbers divisible by 3' do
    @cp.what_is(3).must_equal 'Crackle'
  end

  it 'must return "Pop" for numbers divisible by 5' do
    @cp.what_is(5).must_equal 'Pop'
  end

  it 'must return "CracklePop" for 15' do
    @cp.what_is(15).must_equal 'CracklePop'
  end

  it 'must return 2 for 2' do
    @cp.what_is(2).must_equal 2
  end

  it 'must output 1 - 100 according to rules' do
    -> { puts @cp.to_s }.must_output "1\n2\nCrackle\n4\nPop\nCrackle\n7\n8\nCrackle\nPop\n11\nCrackle\n13\n14\nCracklePop\n16\n17\nCrackle\n19\nPop\nCrackle\n22\n23\nCrackle\nPop\n26\nCrackle\n28\n29\nCracklePop\n31\n32\nCrackle\n34\nPop\nCrackle\n37\n38\nCrackle\nPop\n41\nCrackle\n43\n44\nCracklePop\n46\n47\nCrackle\n49\nPop\nCrackle\n52\n53\nCrackle\nPop\n56\nCrackle\n58\n59\nCracklePop\n61\n62\nCrackle\n64\nPop\nCrackle\n67\n68\nCrackle\nPop\n71\nCrackle\n73\n74\nCracklePop\n76\n77\nCrackle\n79\nPop\nCrackle\n82\n83\nCrackle\nPop\n86\nCrackle\n88\n89\nCracklePop\n91\n92\nCrackle\n94\nPop\nCrackle\n97\n98\nCrackle\nPop\n"
  end

  it 'must output -15 - 15 according to rules' do
    -> { puts @cp.to_s(-15..15) }.must_output "CracklePop\n-14\n-13\nCrackle\n-11\nPop\nCrackle\n-8\n-7\nCrackle\nPop\n-4\nCrackle\n-2\n-1\nCracklePop\n1\n2\nCrackle\n4\nPop\nCrackle\n7\n8\nCrackle\nPop\n11\nCrackle\n13\n14\nCracklePop\n"
  end

  it 'must perform custom rules' do
    -> { puts CracklePop.new( 7 => 'Snap').to_s(5..8) }.must_output "5\n6\nSnap\n8\n"
  end
end
```

```
%w[bin lib test].each do |path|
  $:.unshift('./' + path)
end
```