```
puts "DPLL in Ruby..."

class DPLL

  attr_accessor :formula
  def initialize(formula)
    self.formula = formula
  end

  def solve
    mappings = {}
    variables(formula).each{|variable| mappings[variable] = nil}
    solve_iter(clone_formula(formula), mappings)
  end

  def clone_formula(formula)
    formula.map(&:clone)
  end

  def variables(formula)
    formula.flatten.map!{|var| var.abs}.uniq
  end

  def solve_iter(formula, mappings)
    return false if contains_empty_clause?(formula)
    return "Solution found: #{mappings.inspect}" if formula.empty?
    remove_unit_clauses(formula, mappings)
    vars = variables(formula)
    vars.each do |variable|
      try = clone_formula(formula)
      try_mappings = mappings.clone
      try_mappings[variable] = true
      propagate_variable(variable, try, try_mappings)
      result = solve_iter(try, try_mappings)
      if result
        return result
      else
        try = clone_formula(formula)
        try_mappings = mappings.clone
        try_mappings[variable] = false
        propagate_variable(variable, try, try_mappings)
        return solve_iter(try, try_mappings)
      end
    end
    solve_iter(formula, mappings)
  end
```

```
  def contains_empty_clause?(formula)
    formula.any?{|clause| clause.empty?}
  end

  def remove_unit_clauses(formula, mappings)
    while unit_clause = formula.find{|clause| clause.size == 1}
      formula.delete unit_clause
      variable = unit_clause.first
      mappings[variable.abs] = (variable > 0 ? true : false)
      propagate_variable(variable.abs, formula, mappings)
    end
  end

  def propagate_variable(variable, formula, mappings)
    value = mappings[variable]
    formula.each do |clause|
      while index = clause.index(variable)
        clause[index] = value
      end
      while index = clause.index(-variable)
        clause[index] = !value
      end
    end
    reduce_clauses(formula)
  end

  def reduce_clauses(formula)
    formula.map!{|clause| clause.delete(false) || clause}
    formula.delete_if{|clause| clause.index(true)}
  end
end

puts DPLL.new([[1 -2],[2,-3,4], [1, -4],[5,6],[7,-9],[5,8,9]]).solve
```