Assignment 2: Satisfiability (SAT) solver

Why SAT?

- Logical inference.
- The theory of NP-completeness says that every problem in NP can be reduced to one another. Come up with efficient algorithms for just one.
- Many real-world algorithms use a SAT solver at their core:
  - Hardware/software verification.
  - Logistics.
  - Mechanical/medical diagnosis.
  - Automated reasoning.
- There are annual competitions and journals devoted just to SAT: http://www.satcompetition.org/

SAT problem

- Take as input a propositional statement in CNF format:
  - a -b -c
  - c b
  - -a
- Find a satisfying assignment: -a -b c

Backtracking search

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
    return BACKTRACK({ }, csp)

function BACKTRACK(assignment, csp) returns a solution, or failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment then
            add {var = value} to assignment
            inferences ← INFERENCE(csp, var, value)
            if inferences ≠ failure then
                add inferences to assignment
                result ← BACKTRACK(assignment, csp)
                if result ≠ failure then
                    return result
        remove {var = value} and inferences from assignment
return failure
```

DPLL algorithm

function DPLL-SATISFIABLE?(s) **returns** *true* or *false*
  inputs: s, a sentence in propositional logic

  $clauses \leftarrow$ the set of clauses in the CNF representation of s
  $symbols \leftarrow$ a list of the proposition symbols in s
  return DPLL($clauses, symbols, \{\ \}$)

---

function DPLL($clauses, symbols, model$) **returns** *true* or *false*

  if every clause in $clauses$ is true in $model$ **then return** *true*
  if some clause in $clauses$ is false in $model$ **then return** *false*
  $P, value \leftarrow$ FIND-PURE-SYMBOL($symbols, clauses, model$)
  if P is non-null **then return** DPLL($clauses, symbols - P, model \cup \{P{=}value\}$)
  $P, value \leftarrow$ FIND-UNIT-CLAUSE($clauses, model$)
  if P is non-null **then return** DPLL($clauses, symbols - P, model \cup \{P{=}value\}$)
  $P \leftarrow$ FIRST($symbols$); $rest \leftarrow$ REST($symbols$)
  return DPLL($clauses, rest, model \cup \{P{=}true\}$) **or**
        DPLL($clauses, rest, model \cup \{P{=}false\}$))

Bonus problem

- Implement the best SAT solver you can, using any algorithm, programming language, or compute infrastructure.