CS402
Introduction to Logic in Computer Science

Coursework 1: Normal Forms, Validity, and Satisfiability
Due on 18:00, 3rd April 2018

1 Conjunctive Normal Form

Write a program that takes a single propositional formula in Polish Notation as input, and prints out the following three results: 1) its Conjunctive Normal Form in Polish Notation, 2) its CNF form in infix notation (with appropriate use of parentheses), and 3) the decision on the validity of the formula.

- Assume that the tokens (i.e. operators and literals) in the input are separated by a single space. This applies to negation operator as well (i.e. `- a1` ($\circ$), `-a1` ($\times$)).
- The program needs to consider only the operators below; literals can be any alphanumeric words.
- Take the input from the command-line arguments.
- Use the following symbols for operators in both input and output.

Operator	Symbol	Operator	Symbol
AND	&	Implication	>
OR	|	Reverse Implication	<
NOT	-	Equivalence	=

For example,

```
$ cnf > & - p q & p > r q  //φ = (¬p ∧ q) → (p ∧ (r → q))
& | p | - q p | p | - q | - r q //φ' = (p ∨ ¬q ∨ p) ∧ (p ∨ ¬q ∨ ¬r ∨ q)
(p | - q | p) & (p | - q | - r | q) //φ' = (p ∨ ¬q ∨ p) ∧ (p ∨ ¬q ∨ ¬r ∨ q)
Not Valid
$ _
```

2 Nonogram as SAT

Write a MiniSAT-based solver for Nonogram(`https://en.wikipedia.org/wiki/Nonogram`). It should take a single puzzle as an input, formatted in the CWD format as described below. Then it should print out the solution, using `.` to denote a black space, and `#` to denote a filled space. Use the CNF converter you developed for the first task if required.

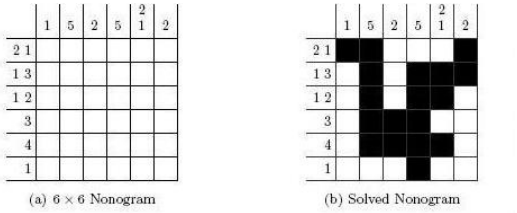

Figure 1: A small Nonogram example

CWD is a plaintext format; each of the first two lines contains the number of rows and columns, respectively. The remaining lines contain clues for all the rows, followed by all the columns. For example, the CWD file for the puzzle in Figure 1 is as follows:

```
6
6
2 1
1 3
1 2
3
4
1
1
5
2
5
2 1
2
```

Figure 2: Example CWD format

3 Deliverables

Each person should submit the following deliverables by the submission deadline:

- **Implementation**: code for two problems, self-contained in separate directories (see below).
- **Report**: include a written report that contains detailed descriptions of how you approached the problems. For the Nonogram solver, clearly outline how you encoded the puzzle into SAT. There is no page limit.

For ease of marking, follow the following directory structure, and submit a zip file containing the top level directory, through KLMS.

```
[your student number]
├── report.pdf ..... Your report documenting both implementations
├── cnf ..... CNF converter and dependencies
└── nonogram ..... Nonogram solver and dependencies
```

4 Guidelines

- You are free to choose any programming language.
- **BUT we expect solutions for both tasks to run straight out of the box.** Your submission should be self-contained, and it should run on Unix-like systems (apologies to Windows users, but `minisat` depends on `cygwin` on Windows, which makes it *harder than NP-hard* to mark these things). You can assume that `minisat` is available on path.
- Avoid printing anything other than the final solution, as it will make marking harder.
- Plagiarism will only result in zero marks; do your own work.
- Document your implementation as best as you can; whatever you do not describe, you risk having it being overlooked by the markers!

5 Evaluation and Competition

The following marking criteria apply:

- Effectiveness and efficiency of the CNF converter: 30%
- Effectiveness and efficiency of the Nonogram solver: 30%
- Quality of both the code and the report: 40%

Submissions will be marked using private inputs and puzzles of considerable size. Outstanding submissions will win prizes (not related to grading, of course).