Assignment 3 (*Marks: 7, Due date: Oct. 11*) **COL703, Aug-Nov 2022**

Please note that there will be zero tolerance for dishonest means like copying solutions from others, and even letting others copy your solution, in any assignment/quiz/exam. If you are found indulging in such an activity, your answer-paper/code will not be evaluated and your participation/submission will not be counted. Second-time offenders will be summarily awarded an F grade. The onus will be on the supposed offender to prove his or her innocence.

---

1. A *renamable Horn formula* is a CNF formula that can be turned into a Horn formula by negating (all occurrences of) some of its variables. For example,

$$(p_1 \vee \neg p_2 \vee \neg p_3) \wedge (p_2 \vee p_3) \wedge (\neg p_1)$$

can be turned into a Horn formula by negating p_1 and p_2.

Given a CNF formula F, it is in fact possible to derive a 2-CNF formula G such that G is satisfiable if and only if F is a renamable Horn formula. Moreover, you can derive a renaming (that turns F into a Horn formula) from a satisfying assignment for G.

Your task is to write a Python program that takes a CNF formula F, in DIMACS format, and also take a positive integer n ($1 \leq n \leq 4$) as a command-line argument, and does the following (for different values of n).

Let us take the following formula as our *sample input*:

```
c CNF formula (p1 ∨ !p2 ∨ !p3) ∧ (p2 ∨ p3) ∧ (!p1)
p cnf 3 3
1 -2 -3 0
2 3 0
-1 0
```

(a) **[1 marks]** When $n = 1$, your program should check whether the given formula is already a Horn formula or not, and output `already horn` and `not horn` accordingly.

Sample output: `not horn`

(b) **[2 marks]** When $n = 2$, your program should output a 2-CNF formula G, also in DIMACS format, such that G is satisfiable if and only if F is a renamable Horn formula.

Sample output:

```
c 2-CNF formula which is sat iff input is renamable Horn
p cnf 3 4
-2 -3 0
2 3 0
1 -3 0
1 -2 0
```

(c) **[2 marks]** When $n = 3$, your program should check whether F is a renamable Horn formula or not (in other words, whether the G that you would have output in the previous case is satisfiable or not[1]). If F is already a Horn formula, the program should output `already horn`. Otherwise, it should output `renamable` or `not renamable` accordingly.

Sample output: `renamable`

(d) **[1 marks]** When $n = 4$, your program should output a way to do the renaming of F. Essentially, you just need to output all the variables (space separated, sorted in ascending order, all in one line) that need to be negated. If the formula is not a renamable Horn formula or if it is already a Horn formula to begin with, you should output `not renamable` or `already horn` accordingly.

Sample output: `1 2`

[1 marks] Submit two CNF formulas as input files in DIMACS format, each having 4 or more variables and 4 or more clauses, none of them being Horn formulas to begin with, such that one is a renamable Horn formula while the other is not.

Submission instructions: Submit three files: `assign3.py`, `input1.cnf`, and `input2.cnf`. The first one should have your code (in the template given here: `https://kumarmadhukar.github.io/courses/assign3.py`), and the next two should have a CNF formula each, in DIMACS format, as asked in the problem statement above. Please also note that your program must not print anything other than the exact output as mentioned above.

[1] It might be interesting to do this by resolution. But you are free to implement any algorithm of your choice.