Context: Satisfiability [easy]. If this/similar appears as regular HW, it is not required. A boolean formula is satisfiable if there exists some variable assignment that makes the formula evaluate to true. Namely, a boolean formula is satisfiable if there is some row of the truth table that comes out true. Determining whether an arbitrary boolean formula is satisfiable is called the *Satisfiability Problem*. There is no known efficient solution to this problem, in fact, an efficient solution would earn you a million dollar prize. While this is hard problem in computer science, not all instances of the problem are hard, in fact, determining satisfiability for some types of boolean formulae is easy.

i. First, let's consider why this would be hard. If you knew nothing about a given boolean formula other than that it had n variables, how large is the truth table you would need to construct? Please indicate the number of columns and rows as a function of n

ii. Now consider the following 100 variable formula.

$$x_1 \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_3 \vee x_4) \wedge \ldots \wedge (\neg x_{99} \vee x_{100})$$

Without constructing a truth table, how many satisfying assignments does this formula have, explain your answer.

iii Now consider an arbitrary 3-DNF formula with 100 variables and 200 clauses. 3-DNF means that the formula is in disjunctive normal form and each clause has three literals. (A literal is the instantiation of the variable in the formula, so for x, $\neg x$ or x.) An example might be something like:

$$(\neg x_1 \wedge x_3 \wedge x_{10}) \vee (\neg x_3 \wedge x_{15} \wedge \neg x_{84}) \vee (x_{17} \wedge \neg x_{37} \wedge x_{48}) \vee \ldots \vee (\neg x_{87} \wedge \neg x_{95} \wedge x_{100})$$

What is the largest size truth table needed to solve this problem. What is the maximum number of such truth tables needed to determine satisfiabilty.

part B: 2CNF-SAT [required]. The 2CNF-SAT instance is a boolean CNF formula with 2 variables in each clause, "OR" inside clauses, "AND" between clauses. There are m boolean variables $x_1, x_2, ..., x_m$) and n clauses $C_1, C_2, ..., C_n$). Every variable and its negation appears in at least one clause. Such formula is given as input in format redundantly :
- for each variable there is a list of clauses containing it
- for each clause there there are 2 variables
For example the formula $(x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_3)$ will be given as:
$m = 3, n = 4$
$x_1 : C_1$
$\neg x_1 : C_3$
$x_2 : C_2$
$\neg x_2 : C_1, C_4$
$x_3 : C_2, C_3$
$\neg x_3 : C_4$
$C_1 : x_1, \neg x_2$
$C_2 : x_2, x_2$
$C_3 : \neg x_1, x_3$
$C_4 : \neg x_2, \neg x_3$

Your task is to design a strategy that determines, for a given formula, the boolean assignments for the variables such that all clauses are satisfied, thus the formula is true (if more such assignments are possible, you only need to output one). If no such assignment is possible, output "FALSE".

As established inpart A, there are 2^m possible assignments for the variable set. So if one were to build the truth table and "brute force" search all rows/assignments until one works, it would take exponential time — not good! Instead: do trial and error, but in a smart way that only tries at most $2 * m^2$ boolean assignments.

Your strategy can be pseudocode, or you can informally describe a procedure with bullets and English statements. You can write in your procedure statements like
* $x = x_1$

* foreach C containing variable x {
- - - -
}
* C= next clause, or C = next clause containing x

* loop C through all clauses that contain x or $\neg x$

* for each $x \in C$ {
- - - -
}
* y = the other variable in clause C, other than x or $\neg x$

part C) optional, no credit [very hard] Can you adapt your strategy from part B to work on 3SAT-CNF formula, where each clause has exactly 3 variables? Why or why not? How many trial assignments of the variables your program has to make to determine the satisfiable assignment?

Example 3SAT formula

$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_1)$

$m = 4, n = 4$

$x_1 : C_1$

$\neg x_1 : C_3, C_4$

$x_2 : C_2$

$\neg x_2 : C_1, C_4$

$x_3 : C_1, C_2, C_3$

$\neg x_3 : C_4$

$x_4 : C_2$

$\neg x_4 : C_3$

$C_1 : x_1, \neg x_2, x_3$

$C_2 : x_2, x_3, x_4$

$C_3 : \neg x_1, x_3, \neg x_4$

$C_4 : \neg x_2, \neg x_3, \neg x_1$