COT 3100 Program #2 Spring 2017
Assigned: 4/10/2017
Check WebCourses for due date

Note: This program is purely optional. 5% of the course grade is determined from either attendance in recitation OR two programs. The maximum of the two options will be what each student receives for their grade.

General Program Directions (to be followed for all five programs)

Turn in a single source file, in Python, C or Java, with the name designated in the note at the bottom of this page that solves the problem described below. Please read all of your input from standard in and output to standard out. To test your program with input files, please pipe the input file into your program and pipe the output to another file. If you don't know how to do this, please see a TA to describe this process to you.

General Grading Criteria

Your submissions for all programs will be graded based on points in these two categories:

1) Execution Points
2) Coding Style Points

The first criteria will 80% of the program grade. It will simply be based on the number of test cases your program produces correct output on within the designated time limit. (No test cases after a crash will be run to see if your program would have passed them.)

The second criteria will typically be 20% of the program grade. It will look very similar to my Computer Science I style criteria. Use good variable names, indent properly, use reasonable white space, comment appropriately, as well as good coding style (use of methods/functions when appropriate, reasonable problem breakdown and so forth.)

Deliverables

A single source file, named ***threesat.py***, ***threesat.c***, ***threesat.java*** that solves the program stated above, using the input and output formats stated above, using standard input and standard output. The file should be submitted via WebCourses by the due date and time stated in WebCourses.

The Problem: 3-CNF-SAT

A classic problem in computer science is satisfiability. A boolean formula is simply an expression with boolean variables and boolean operators. 3 conjunctive normal form (3-CNF), are boolean formulas in a fixed structure. The overall structure has a number of clauses, each of which are "anded" together. Using the appropriate notation all valid boolean formulas in 3-CNF form can be expressed as:

$$\bigwedge_{i=1}^{n} C_i$$

where C_i represents the i[th] clause in the boolean formula. Each clause itself, must be of the form

$$x \vee y \vee z$$

where x, y, z and are variables or the negation of variables. The following is a valid boolean formula in 3-CNF form with variables p, q and r:

$$(p \vee \bar{q} \vee r) \wedge (\bar{p} \vee q \vee \bar{r}) \wedge (\bar{p} \vee q \vee r) \wedge (\bar{q} \vee \bar{q} \vee \bar{r}) \wedge (p \vee p \vee \bar{r})$$

The 3-CNF-SAT problem is defined as follows: given a boolean formula in 3-CNF form, determine whether or not there exist a setting of the variables such that when that particular setting is plugged into the expression, the expression evaluates to true.

For the expression above, if we set $p = True, q = True, r = False$, then the whole equation evaluates to true. To see this, note that p makes the first clause true, q makes the second clause true, q makes the third clause true, $\bar{r}$ makes the fourth clause true and p makes the fifth clause true.

Notice that if we set $p = True, q = False, r = False$, then the corresponding boolean formula would be false because the third clause would be false, since for this particular truth setting, all three variables in the clause equal false. Note that if one clause is false, the whole formula is false.

Thus, a boolean equation is satisfiable so long as ***there exists*** some variable setting that makes it true. It doesn't matter that some variable settings make the formula false. A boolean formula is NOT satisfiable if there does NOT exist a truth setting that makes the boolean formula true.

This problem is an NP-Complete problem. That means that no one knows of a "fast" way to solve it. Rather, most known approaches to solve this problem involve some sort of brute force check where we simply try all possible settings of the variables.

For this program, that is what you're supposed to do: read in a boolean formula in 3-CNF form and determine whether or not the formula is satisfiable via brute force. Namely, your program should simply try all 2^n possible settings of the variables, where n represents the number of different variables in the given expression and see if any of them make the formula true.

Input Format

The first line of the input file will contain a single positive integer, *n*, representing the number of input cases to process. The input cases follow. The first line of each input case have two space separated positive integers, v ($1 \leq v \leq 15$) and c ($1 \leq c \leq 1000$), representing the number of variables and clauses for that input case, respectively. The variables will be numbered 1 through v and the negation of a particular variable will be denoted with the negative of the variable number. The following c lines contain each of the clauses. The i^{th} of these lines will contain three space separated integers representing variables or their negations that comprise the i^{th} clause.

Output Format

For each input case, output the string "yes" on a line by itself if the formula is satisfiable, or "no" if it is not.

Sample Input

```
2
3 5
1 -2 3
-1 2 -3
-1 2 3
-2 -2 -3
1 1 -3
4 10
1 2 3
1 2 -3
1 -2 3
1 -2 -3
-1 2 3
-1 2 -3
-1 -2 3
-1 -2 -3
1 1 4
1 1 -4
```

Sample Output

```
yes
no
```