

## Problem 155: Three's Company

Difficulty: Hard

Author: Louis Ronat, Denver, Colorado, United States

Originally Published: Code Quest 2021

### Problem Background

Computer programs are complicated. However, analyzing just *how* complicated computer programs are - determining their computational complexity - is a fundamental topic in computer science. Different classes of computational complexity exist, and one of the biggest unproven questions in complexity analysis deals with the class known as NP problems. An NP problem is one where it is believed that an algorithm solving the problem must take an amount of time that grows faster than a polynomial compared to the size of the input. That is, if  $a$  is a constant and  $n$  is the size of the input, the time needed to solve the problem increases faster than the value of  $n^a$ . Many of the most difficult Lockheed Martin Code Quest problems fall into this category.

As complicated as these problems are to solve, verifying that a particular solution is correct is not as difficult. NP problems have a property where verifying a provided solution can be completed in polynomial time. Today, you'll be verifying one such NP problem, known as 3-satisfiability or 3-SAT.

### Problem Description

3-SAT problems start with a logical statement grouped in a specific manner. The statement includes two or more "triplets" of Boolean inputs, joined with a logical OR statement. One such triplet is shown below:

**A or B or C**

Each triplet is then joined together with a logical AND to form the full logical statement. For example:

**(A or B or C) and (!D or !E or !F)**

Note that the ! character appears here as a negation; !D is true when D is false.

Solving a 3-SAT problem requires answering a question: "Will this statement ever be true?" The statement given above will be true if at least one of the inputs A, B, or C is true, and at least one of the inputs D, E, and F is false. As long as some combination of possible inputs matches those criteria, the answer in this case is yes; if no such combination of inputs is valid, the answer is no.

For example, consider if that statement were placed in an if statement in your programming language. The 3-SAT problem's answer is "yes" if there is some way to execute the code inside that if block. If there is no way to access that code, the answer is "no."

A 3-SAT can have more than two triplets; in each case, the members within a triplet are joined using a logical OR, and each triplet is joined with a logical AND to form the complete statement.

Your team must create a program that can build a given 3-SAT statement and test it against several inputs to determine if the problem is solvable. Be careful; the true/false inputs will be provided in alphabetical order, so you'll need to be sure to correctly map them to the variables in the triplets.

## Sample Input

The first line of your program's input, received from the standard input channel, will contain a positive integer representing the number of test cases. Each test case will include:

- A line containing two positive integers separated by spaces:
  - The first integer, *T*, represents the number of triplets in the 3-SAT statement. *T* will be greater than or equal to 2.
  - The second integer, *I*, represents the number of inputs to test against the final statement.
- *T* lines containing three input values, separated by spaces, representing the triplets that will form the 3-SAT statement. Each line represents a single triplet, and each input value will consist of a single uppercase letter. Letters may be prefaced with an exclamation point (!), indicating that that input should be negated when added to the 3-SAT statement. Within each test case, the letters used will start with A and will not skip any letters, however the letters may not be presented in alphabetical order.
- *I* lines, each containing 3*T* 0's or 1's separated by spaces, representing the input values to be tested against the 3-SAT statement. A value of 0 represents 'false', and a value of 1 represents 'true'. Values are presented in alphabetical order; the first value represents the value of A; the second, the value of B, and so on.

```
1
2 4
A B C
!D !E !F
1 0 0 1 1 0
1 1 1 1 1 1
0 0 0 0 0 0
1 1 1 0 0 0
```

## Sample Output

For each test case, your program must output one line for each set of values to be tested, containing the word "TRUE" if the 3-SAT statement valuated to 'true' with the given values, or "FALSE" otherwise.

From Lockheed Martin Code Quest™ Academy - [www.lmcodequestacademy.com](http://www.lmcodequestacademy.com)

TRUE  
FALSE  
FALSE  
TRUE