acm Greater New York Programming Contest 2003 IBM event sponsor

E • It’s Logical

Problem

At the University of Kentucky, they build a lot of high-performance computer hardware and software, often using one supercomputer to design the next. One of the most fundamental computer design problems is logic optimization: making sure that the optimized logic still computes the same function as the original design.

For this problem, your program will be given two logic expressions to compare for logical equivalence.

Input

The first line of the input consists of a positive integer *n*, which is the number of datasets (lines) that follow. Each dataset consists of a single line containing the two input expressions to be tested. The input expressions consist of any of 26 variables named *a*-*z*, the binary operators |, &, ^, (OR, AND and XOR respectively), the unary ~ (NOT), and parenthesis. The expressions should be evaluated ignoring all other characters and with operator precedence as in the C language (parenthesis, ~, &, ^, |). The two expressions will be input in sequence and it is up to your program to determine where one expression ends and the next begins.

Most logic manipulation programs would convert each expression into a normal form and check if the two normalized expressions are identical. Fortunately for you, each expression will consist of no more than 100 operations using no more than 10 different variables. For that many cases, you can test for equivalence by simply evaluating the two input expressions for all possible inputs and comparing the results.

Output

For each data set, print:

```
Data set N: Equivalent
```

if the expressions produce the same result, or:

```
Data set N: Different
```

if they produce different results. Of course *N* should be replaced by the data set number.

Example

Input	**Output**
`3`	`Data set 1: Different`
`a ^b&(b|a)~b^ a`	`Data set 2: Equivalent`
`a^b&(b|a)(a^(b&(b|a)))`	`Data set 3: Equivalent`
`~~~~z~~z`	