A software company is very much concerned that its software engineers do not write equivalent procedures into the new version of its main product BEATBIT. The BEATBIT system is written in the assembly language BITE, recently introduced by MacroSoft, in its .DOT Framework. So far, no one has invented a more powerful language to program complex digital circuits. The BITE assembly language operates on a stack of bits, and is defined by the following four kinds of instructions:

> *label* `BRTRUE` *destlabel*
>
> Pops a bit from the top of the stack, and tests it. If the value is 1 continues execution at the instruction with label *destlabel*. If the value is 0 continues at the next program instruction.
>
> *label* `JMP` *destlabel*
>
> Continues execution at the instruction with label *destlabel*.
>
> *label* `RET1`
>
> Stops execution and returns **1**.
>
> *label* `RET0`
>
> Stops execution and returns **0**.

Here, *label* and *destlabel* are positive integers. A n-ary procedure of BITE is a sequence of instructions that expects N bit values on the stack as input, and produces one bit value, as the result of a `RET0` or `RET1` instruction. The instructions in the sequence are always labelled in increasing label sequence, and it is known that, for every possible input, the procedures always terminate. The program starts at the instruction with the lowest label.

Write a program that checks whether two BITE procedures compute the same boolean function for any sequence of the values, stored in the stack, provided as input.

Input

Input consists of multiple test cases, each of them as described below. The first line of the input contains the number of test cases.

There is a blank line before each dataset.

A positive integer P in a single line followed by a sequence of P pairs of BITE procedures. Each pair of BITE procedures is preceded by the number of bits expected in the stack (the arity of the procedures), represented by a positive integer, not greater than 128, in a single line. Next, for each BITE procedure there is a sequence of lines, each one containing a BITE instruction, and terminated by a single line containing END.

Programs do not have more than 10,000 lines of code.

Output

For each dataset, a sequence of lines, the i-th line containing either '`1`' or '`0`' depending on whether the ith pair of BITE procedures in the input compute the same boolean function or not.

Print a blank line between datasets.

Sample Input

```
1

2
2
10 BRTRUE 30
20 RET0
30 BRTRUE 50
40 RET0
50 RET1
END
20 BRTRUE 50
30 BRTRUE 40
40 RET0
50 BRTRUE 70
60 JMP 40
70 RET1
END
1
10 BRTRUE 30
20 RET0
30 RET1
END
50 RET0
END
```

Sample Output

```
1
0
```