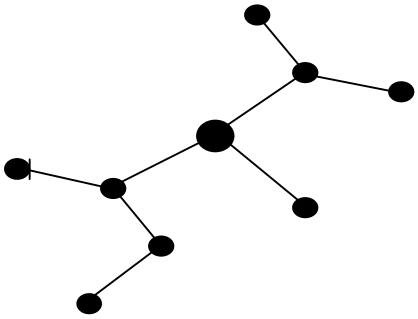
Problem E

Subway tree systems

*Source code: subway.**

Some major cities have subway systems in the form of a tree, i.e. between any pair of stations, there is one and only one way of going by subway. Moreover, most of these cities have a unique central station. Imagine you are a tourist in one of these cities and you want to explore all of the subway system. You start at the central station and pick a subway line at random and jump aboard the subway car. Every time you arrive at a station, you pick one of the subway lines you have not yet travelled on. If there is none left to explore at your current station, you take the subway line back on which you first came to the station, until you eventually have travelled along all of the lines twice, once for each direction. At that point you are back at the central station. Afterwards, all you remember of the order of your exploration is whether you went further away from the central station or back towards it at any given time, i.e. you could encode your tour as a binary string, where 0 encodes taking a subway line getting you one station further away from the central station, and 1 encodes getting you one station closer to the central station.



```
0010011101001011
0100011011001011
0100101100100111
...
```

Figure 1. To the left: A subway tree system. The larger dot is the central station. To the right: Three out of several possible encodings of exploration tours for the subway system.

Input

On the first line of input is a single positive integer *n*, telling the number of test scenarios to follow. Each test scenario consists of two lines, each containing a string of the characters '0' and '1' of length at most 3000, both describing a correct exploration tour of a subway tree system.

Output

For each test scenario, output one line containing the text “same” if the two strings may encode exploration tours of the same subway tree system, or the text “different” if the two strings cannot be exploration tours of the same subway tree system.

Example input:

```
2
0010011101001011
0100011011001011
0100101100100111
0011000111010101
```

Example output:

```
same
different
```