usermind Take-home Coding Problem

Dear candidate,

We're very grateful for your interest in a software development position with usermind. We pride ourselves in building a strong engineering team with a great culture and great development chops. We know that the interview process is a tricky business so we continually adjust it to better identify great engineers while allowing everyone the best opportunity to showcase their abilities. This take home coding problem is an attempt to remove the pressure of an in-person interview and the artificiality of a whiteboard. We hope you'll enjoy writing some interesting code as much as we've enjoyed coming up with the problem and solving it ourselves.

Solve the problem in any language you like but make it professional as you would in for a production system. This is an opportunity to do your best work without time constraint and with your favorite tools at your disposal.

Have fun and good luck!

usermind engineering team

Brilliant DeTECHtive

You're a detective. For each crime you're investigating, you've got a number of witnesses that each remember the order of events they witnessed. These witnesses are trustworthy. However, any single witness might not have witnessed all events. For instance:

Witness Alice remembers:	["shouting", "fight", "fleeing"]
Witness Bob remembers:	["fight", "gunshot", "panic", "fleeing"]
Witness Craig remembers:	["anger", "shouting"]

As the detective, we have to put together a timeline of events, in the order they occurred. The longer the resulting timeline is, the better. If all witnesses present a single consistent sequence of events or if at least parts of them can be combined to create a longer timeline then the probability of a successful conviction goes up. Therefore, it's important to merge their timelines to the maximum degree possible. However, there ordering of events must be absolutely correct or else the judge will throw our case out so in case there are events in those timelines that cannot be strictly ordered, it is better to present multiple separate timelines.

Therefore, for each case, you have to determine how much of the events can be merged:

- if the all witnesses remember events in an fully consistent manner then present a single merged timeline,

- if some of the events they remember allow to combine some of the timelines or if some of them can be extended without fully merging them then present multiple timelines with events merged across them to the degree possible
- if none of the events allow combining or extending any of the timelines then present the original unmodified timelines.

For the above case, we can condense all witness accounts down to a single timeline:

["anger", "shouting", "fight", "gunshot", "panic", "fleeing"]

In other cases/crimes, we may have to present multiple possible timelines:

Witness Dan: ["pouring gas", "laughing", "lighting match", "fire"]

Witness Ed: ["buying gas", "pouring gas", "crying", "fire", "smoke"]

We can't tell whether the arsonist was crying before or after laughing / lighting the match, so we'd have to present the prosecution with two possible timelines:

["buying gas", "pouring gas", "laughing", "lighting match", "fire", "smoke"]

["buying gas", "pouring gas", "crying", "fire", "smoke"]

Write a program that, given multiple arrays (eyewitness accounts), produces a minimal set of absolutely ordered, maximally long arrays (timelines) to give to the prosecution.

Examples:

Input	Output

`[`
`["fight", "gunshot", "fleeing"],`
`["gunshot", "falling", "fleeing"]`
`]`
	Merge is possible
	`[`
	`["fight", "gunshot", "falling",`
	`"fleeing"]`
	`]`

`[`
`["shadowy figure", "demands",`
`"scream", "siren"],`
`["shadowy figure", "pointed gun",`
`"scream"]`
`]`
	Partial merge is possible
	`[`
	`["shadowy figure", "demands",`
	`"scream", "siren"],`
	`["shadowy figure", "pointed gun",`
	`"scream", "siren"]`
	`]`

`[`
`["argument", "coverup",`
`"pointing"],`
`["press brief", "scandal",`
`"pointing"],`
	No merge is possible
	`[`
	`["argument", "coverup", "pointing"],`
	`["press brief", "scandal",`
	`"pointing"],`
	`["argument", "bribe"]`

Input	Output
["argument", "bribe"]]	]

Coding Style

Your code should be readable and clearly formatted. It should demonstrate general software engineering best practices, such as error handling and unit tests. Write code you'd feel comfortable having your colleagues maintain, and use this chance to show off your programming prowess! Please ask us if you have any questions.

Programming Execution

Your program should feature a driver that accepts a file name argument, where the file contents are a single test case like the examples above.

For example, if you write your program in python, your driver could be launched in the following way:

```
python yourDetechtiveSolver.py testcase_merge_possible.json
```

testcase_merge_possible.json contents:

```
[
["fight", "gunshot", "fleeing"],
["gunshot", "falling", "fleeing"]
]
```

You can output the results over standard output, formatted as shown in the output examples above.