Вам дано описание наследования классов исключений в следующем формате.
<имя исключения 1> : <имя исключения 2> <имя исключения 3> ... <имя исключения k>
Это означает, что **исключение 1** наследуется от **исключения 2**, **исключения 3**, и т. д.

Или эквивалентно записи:

```
class Error1(Error2, Error3 ... ErrorK):
    pass
```

Антон написал код, который выглядит следующим образом.

```
try:
   foo()
except <имя 1>:
   print("<имя 1>")
except <имя 2>:
   print("<имя 2>")
...
```

Костя посмотрел на этот код и указал Антону на то, что некоторые исключения можно не ловить, так как ранее в коде будет пойман их предок. Но Антон не помнит какие исключения наследуются от каких. Помогите ему выйти из неловкого положения и напишите программу, которая будет определять обработку каких исключений можно удалить из кода.

Важное примечание:
В отличие от предыдущей задачи, типы исключений не созданы.
Создавать классы исключений также не требуется
Мы просим вас промоделировать этот процесс, и понять какие из исключений можно и не ловить, потому что мы уже ранее где-то поймали их предка.

Формат входных данных

В первой строке входных данных содержится целое число **n** - число классов **исключений.**

В следующих **n** строках содержится описание наследования классов. В **i**-й строке указано от каких классов наследуется **i**-й класс. Обратите внимание, что класс может ни от кого не наследоваться. Гарантируется, что класс не наследуется сам от себя (прямо или косвенно), что класс не наследуется явно от одного класса более одного раза.

В следующей строке содержится число **m** - количество обрабатываемых исключений. Следующие **m** строк содержат имена исключений в том порядке, в каком они были написаны у Антона в коде.
Гарантируется, что никакое исключение не обрабатывается дважды.

Формат выходных данных

Выведите в отдельной строке имя каждого исключения, обработку которого можно удалить из кода, не изменив при этом поведение программы. Имена следует выводить в том же порядке, в котором они идут во входных данных.

Пример теста 1

Рассмотрим код

```
try:
    foo()
except ZeroDivision :
    print("ZeroDivision")
except OSError:
    print("OSError")
except ArithmeticError:
    print("ArithmeticError")
except FileNotFoundError:
    print("FileNotFoundError")



...
```

По условию этого теста, Костя посмотрел на этот код, и сказал Антону, что исключение FileNotFoundError можно не ловить, ведь мы уже ловим OSError -- предок FileNotFoundError

---

Sample Input:

```
4
ArithmeticError
ZeroDivisionError : ArithmeticError
```

```
OSError
FileNotFoundError : OSError
4
ZeroDivisionError
OSError
ArithmeticError
FileNotFoundError
```

Sample Output:

```
FileNotFoundError
```

Time Limit: 2 секунды
Memory Limit: 256 MB