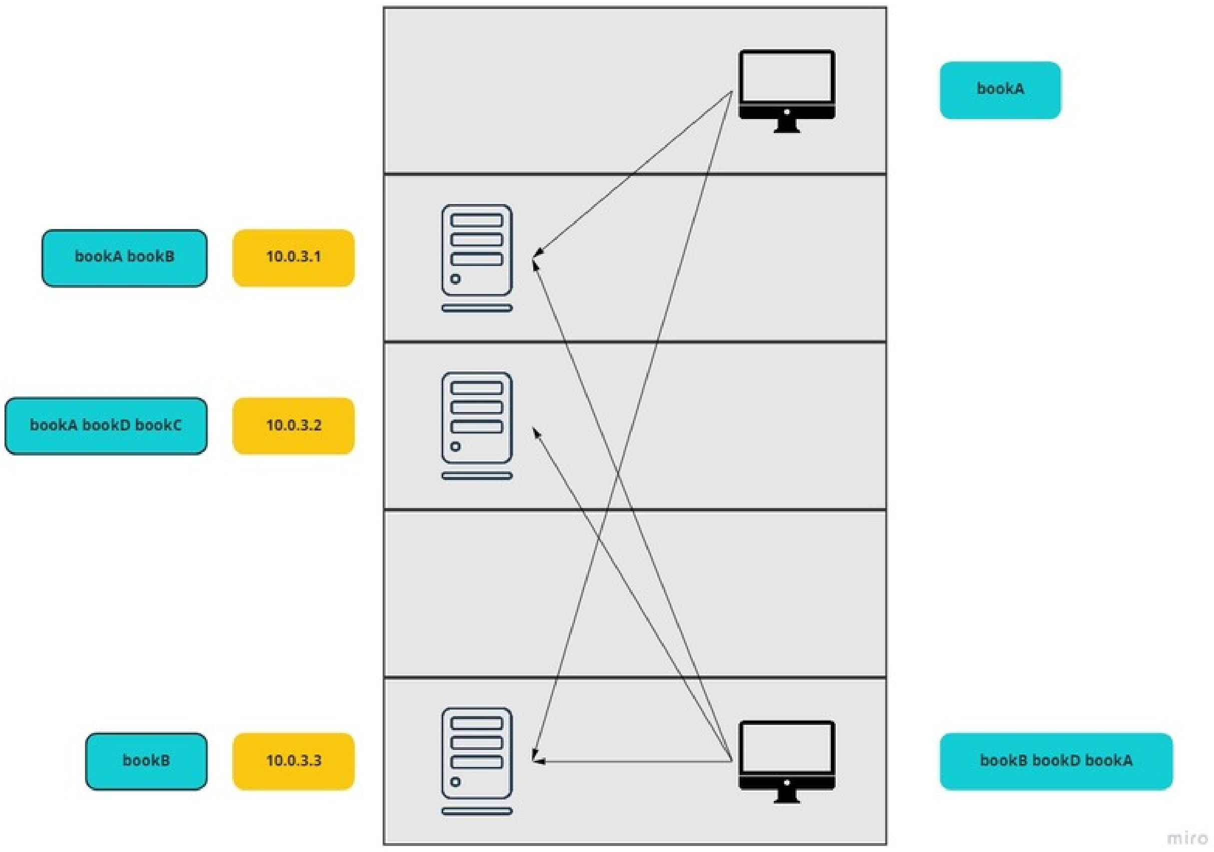
Bepi and His Library Servers

locked

Problem Submissions Discussions

After seeing great success with his library, Bepi wanted to expand its business operation by storing the books in servers. Visitors could access the books through a client PC within the library and it would connect with the various servers scattered within the library building. To improve data redundancy, Bepi used sharding so a book's data could be stored within several different servers. So when a client PC needs to access the data, it would make connections to all servers that have the data shards. The client PC and server could be located in different floors and the data tend to come faster if they're in the same floor.



Bepi wanted to know the effectiveness of his solution. He wanted you to figure out how many connections are made, the order of data that came to the client, and where the data comes from.

Input Format

First line is **N**, the amount of servers. For each **N**,

- [server IP] [amount of data stored] [floor number]
- [data 1] ... [data ...]

Next is **M**, the amount of client PCs. For each **M**

- [amount of needed data] [floor number]
- [data 1] ... [data ...]

Constraints

- 1 ≤ N,M ≤ 1000
- data name consists of alphabets
- 1 ≤ total data ≤ 100

Output Format

For each client PCs, the output differs according whether they got what they need.

If the client found all the needed data:

- print `Client x got what they need`
- followed by the data name and the IPs of all source server separated by a `|`

If the client did not:

- print `As Client x is unable to find the needed data, they went to the competitor instead :(`

The last line is the number of connections made to the servers `Number of connections: y`

Notes:

1. `x` is based on when the client arrives, not necessarily the floor number
2. `y` is the number of connections
3. If there are more than one data that arrives at the same time, order them lexicographically

Sample Input 0

```
3
10.0.3.1 2 4
bookA bookB
10.0.3.2 3 3
bookA bookD bookC
10.0.3.3 1 1
bookB
2
3 1
bookB bookD bookA
1 5
bookB
```

Sample Output 0

```
Client 1 got what they need
bookB 10.0.3.1|10.0.3.3
bookA 10.0.3.1|10.0.3.2
bookD 10.0.3.2
Client 2 got what they need
bookB 10.0.3.1|10.0.3.3
Number of connections: 5
```

Sample Input 1

```
3
10.0.3.1 2 4
bookA bookB
10.0.3.2 3 3
bookA bookD bookC
10.0.3.3 1 1
bookB
1
1 1
bookZ
```

Sample Output 1

```
As Client 1 is unable to find the needed data, they went to the competitor instead :(
Number of connections: 0
```

Sample Input 2

```
2
10.0.3.1 2 6
bookA bookB
10.0.3.2 2 3
bookD bookC
1
3 1
bookA bookD bookC
```

Sample Output 2

```
Client 1 got what they need
bookC 10.0.3.2
bookD 10.0.3.2
bookA 10.0.3.1
Number of connections: 2
```

Explanation 2

bookC and bookD arrives together, while bookD was requested first, bookC is printed ahead due to lexicographical sorting.

bookA arrives late despite being the first one asked, because it was located further away.

Submissions: 0
Max Score: 1

Rate This Challenge:
☆☆☆☆☆

Download problem statement

Download all test cases

Suggest Edits

Collapse

Admin Options

Edit Challenge

View Submissions

C++20

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;


int main() {
    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    return 0;
}
```