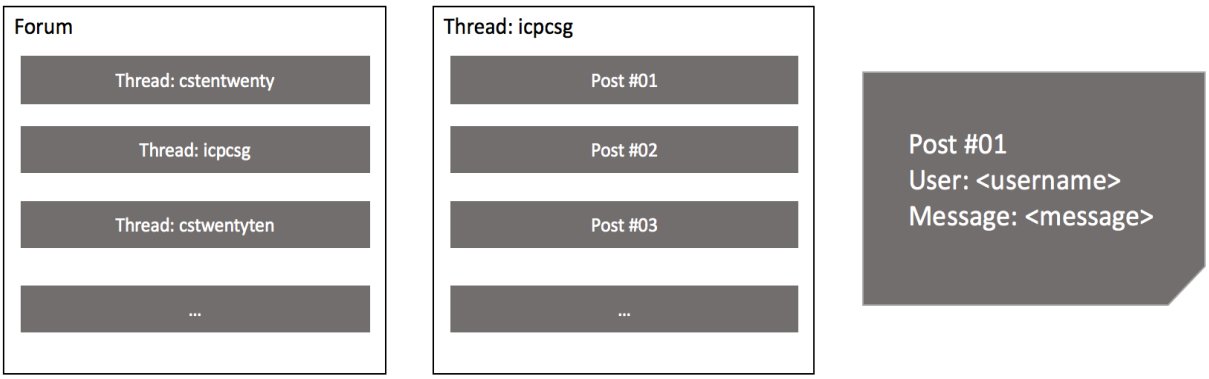
Forum

Steven loves competitive programming. He loves it so much he even writes his own competitive programming book[1]. To further extend his love of competitive programming, Steven wants to create a platform where competitive programmers can interact with one another and share tips and techniques (Steven prefers to use the term "technique" instead of "trick") so that others may benefit from the experience of fellow competitive programmers.

Steven is a skillful coder. However, he does not have enough time to build this forum for competitive programmers. He is too busy maintaining his favorite teaching tool, VisuAlgo[2], which is basically a data structures and algorithms visualization tool. Fortunately, Steven has many students who are willing to help him build this simple forum. This forum, although simple, carries many benefits for competitive programmers worldwide. Therefore, Steven wants this forum to be done quickly and deployed as soon as possible.

As one of his best students, you are approached by Steven to develop this forum system. Steven gives you this high-level overview of the design of the forum:



1. There is only a single forum in which competitive programmers may discuss topics related to competitive programming with each other.
2. In the forum, there are several "threads". A thread is defined as a "sub-forum" in which forum posts are posted. Each thread has its own topic and list of posts.
3. A user can post to any thread within the forum.
4. Once a post is posted, it is not possible to delete it.
5. Each post in a particular thread is numbered in increasing order, starting from 1 for the first post inside that thread.
6. Each post contains the following information:
   - The post number.
   - The username of the user who posted it.
   - The contents of the message in the post.
7. Each user has his or her own unique username.

You have been thinking on how to develop this forum and come up with a brilliant idea. You will create a sophisticated backend forum system which is able to accept queries produced by a query generator based on the actions of a user. This backend system is able to accept the following queries:

[1] Competitive Programming 3: http://cpbook.net/
[2] www.visualgo.net

1. Create a new post from a specific user in a specific thread.
2. Count the total number of posts inside multiple threads.
3. Print the number of posts posted by a particular user.
4. Print the username of the user who is most active in a particular thread (i.e. the user that has posted the most number of posts in a particular thread).
5. Print the content of the post with a particular post number inside a particular thread.

Your job is to create this backend system using your skills as a programmer. Make Steven proud and you shall be rewarded. Good luck!

Input

The first line of input consists of a single number, N (1 <= N <= 20), the number of users registered in the forum. N rows follow. Each of the N rows contains the username of a particular user. The next line contains a single integer T (1 <= T <= 20), the number of threads. T lines follow, each containing the name of a particular thread. The next line contains a single integer, Q (5 <= Q <= 100), the number of queries. Q rows follow. Each of the Q rows contains a single query that the backend system must answer. The queries will follow the following specification:

Query Type Input Format: `<QUERY_TYPE> <APPROPRIATE_PARAMETERS>`

1. `post THREAD_NAME USERNAME MESSAGE`
   Create a new post to the thread THREAD_NAME with the content MESSAGE which is posted by the user USERNAME. If there are no threads with the name THREAD_NAME, print "no such thread". If there are no users with the username USERNAME, print "no such user". If both the tread and username are invalid (i.e. nonexistent), just print "no such thread". If the post is able to be posted, print the message.

2. `count X THREAD_1 THREAD_2 … THREAD_X`
   Print the total number of posts in THREAD_1, THREAD_2, …, and THREAD_X combined. It is guaranteed that all threads provided in the list of threads for this query are valid (i.e. it is already created).

3. `numpost USERNAME`
   Print the number of posts posted by the user with the username USERNAME. If there are no such user, print "no such user".

4. `maxpost THREAD_NAME`
   Print the username of the user who posted the most number of posts in the thread THREAD_NAME. If there are no threads with the name THREAD_NAME, print "no such thread". If the thread is empty (i.e. number of post = 0 for all users) or there are more than one users who posted the most number of posts, print the lexicographically smallest username who has posted the most number of posts).

5. `content THREAD_NAME K`
   Print the content of the post with K as its post number inside the thread THREAD_NAME. If there are no threads with the name THREAD_NAME, print "no such thread'. If there is no post with K as its post number inside the thread THREAD_NAME, print "no such post".

It is guaranteed that all thread names (at most 20 characters) and usernames (at most 10 characters) consist of only lowercase letters ('a' – 'z') and are unique, single words. All messages posted by users can have whitespaces and are at most 50 characters, consisting of purely lowercase letters and / or whitespaces. There might be multiple posts which contain the same message.

Output

Print the result of the query as described in the input format above. The last line of the output should contain a newline character.

Sample Input

```
2
stevenha
zeulb
3
icpcsg
cstentwenty
cstwentyten
10
post icpcsg stevenha hello this is an introductory message
post icpcsg dummyuser hello i am not actually here
post nonexistentthread zeulb this should be an error
post icpcsg zeulb hello my name is stefano
content icpcsg 2
post cstentwenty zeulb tutorial on linked list
numpost zeulb
count 2 icpcsg cstentwenty
post icpcsg stevenha hello i am steven halim
maxpost icpcsg
```

Sample Output

```
hello this is an introductory message
no such user
no such thread
hello my name is stefano
hello my name is stefano
tutorial on linked list
2
3
hello i am steven halim
stevenha
```

Explanation

The first query of the sample input posted the message "`hello this is an introductory message`" to the thread "`icpcsg`" by the user "`stevenha`". The post number is 1 since it is the first post in that thread.

The second query of the sample input is an invalid input since there are no user with the username "`dummyuser`". The third query is a post query by an existing user to a non-existent thread. The fourth query is a valid post query which posts the message "`hello my name is stefano`" to the thread "icpcsg" by the user "`zeulb`" with 2 as its post number. Hence, the next query, which asks for the content of the post numbered "2" in the thread "`icpcsg`" will return "`hello my name is stefano`". The next query posts the first valid message to the thread "`cstentwenty`".

The next query asks for the number of posts that have been posted by the user "`zeulb`", which is 2 (one in `icpcsg` and one in `cstentwenty`). The next query is a `count` query, which counts the number of posts inside 2 threads (as given in the input): `icpcsg` and `cstentwenty`. There are three posts in total, two in `icpcsg` and one in `cstentwenty`. The next query is another valid post query.
The final query is a `maxpost` query, asking who is the user who has posted the most number of posts inside the thread "`icpcsg`". The answer is `stevenha`, who has posted two posts in the thread "`icpcsg`".

Skeleton

You are given the skeleton file **Forum.java**.
Note: You should see a file with the following contents during your sit-in lab when you open the file. If you see an empty file, please go to the correct directory (`ex1`) and see if the skeleton file is located inside.

```
import java.util.*;

public class Forum {
    public static void main(String[] args) {
        //define your main method here
    }
}

class Thread {
    //define appropriate attributes, methods, and constructor
}

class Post {
    //define appropriate attributes, methods, and constructor
}

class User {
    //define appropriate attributes, methods, and constructor
}
```

Notes:

1. You should develop your program in the subdirectory ex1 and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. You only need to modify the skeleton file, that is **you do not need to create a new file for each class.** All code should be inside the file given to you in the ex1 directory.
3. If your algorithm is different from the given skeleton, you are free to write a solution according to your own algorithm. You are free to define your own classes besides the ones given in the skeleton file.
4. You must (and need to) use OOP for this sit-in lab.
5. You are free to define your own methods.
6. Please be reminded that the marking scheme is:

   Input : 10%
   Output : 10%
   Correctness : 50%
   Programming Style : 30%, which consists of:
   - Meaningful comments (pre- and post- conditions, comments inside the code): 10%
   - Modularity (modular programming, proper modifiers [public / private]): 10%
   - Proper Indentation: 5%
   - Meaningful Identifiers (for both method and variable names): 5%