Hypnobox – IT
Technical interview
Software Engineer

Version Control

1.1 – Updating Top Articles endpoint
1.0 – First version of document

1: Football scores

The number of goals achieved by two football teams in matches in a league is given in the form of two lists. For each match of team B, compute the total number of matches of team A where team A has scored *less than or equal to* the number of goals scored by team B in that match.

Example

teamA = [1, 2, 3]

teamB = [2, 4]

Team A has played three matches and has scored *teamA = [1, 2, 3]* goals in each match respectively. Team B has played two matches and has scored *teamB = [2, 4]* goals in each match respectively. For *2* goals scored by team B in its first match, team A has *2* matches with scores *1* and *2*. For *4* goals scored by team B in its second match, team A has *3* matches with scores *1*, *2* and *3*. Hence, the answer is *[2, 3]***.**

Function Description

Complete the function *counts* in the editor below.

counts has the following parameter(s):

int teamA[n]: first array of positive integers

int teamB[m]: second array of positive integers

Return

****int[m]:* an array of *m* positive integers, one *for each teamB[i]* representing the total number of elements from *teamA[j]* satisfying $teamA[j] \leq teamB[i]$ where $0 \leq j < n$ and $0 \leq i < m$, in the given order.

Constraints

- $2 \leq n, m \leq 105$
- $1 \leq teamA[j] \leq 109$, where $0 \leq j < n$.
- $1 \leq teamB[i] \leq 109$, where $0 \leq i < m$.
- **Input Format For Custom Testing**

  Input from stdin will be processed as follows and passed to the function.

  The first line contains an integer *n*, the number of elements in *teamA*.

The next *n* lines each contain an integer describing *teamA[j]* where $0 \leq j < n$.

The next line contains an integer *m*, the number of elements in *teamB*.

The next *m* lines each contain an integer describing *teamB[i]* where $0 \leq i < m$.

- **Sample Case 0**

  Sample Input 0

```
STDIN       Function
-----       --------
4       →   teamA[] size n = 4
1       →   teamA = [1, 4, 2, 4]
4
2
4
2       →   teamB[] size m = 2
3       →   teamB = [3, 5]
5
```

  Sample Output 0

```
2
4
```

  Explanation 0

  Given values are *n = 4*, *teamA = [1, 4, 2, 4]*, *m = 2*, and *teamB = [3, 5]*.

  1. For *teamB[0] = 3*, we have *2* elements in *teamA* (*teamA[0] = 1* and *teamA[2] = 2*) that are ≤ *teamB[0]*.
  2. For *teamB[1] = 5*, we have *4* elements in *teamA* (*teamA[0] = 1*, *teamA[1] = 4*, *teamA[2] = 2*, and *teamA[3] = 4*) that are ≤ *teamB[1]*.

  Thus, the function returns the array *[2, 4]* as the answer.

- **Sample Case 1**

  Sample Input 1

```
STDIN     Function
-----       --------
5       →   teamA[] size n = 5
2       →   teamA = [2, 10, 5, 4, 8]
10
5
4
8
4      →   teamB[] size m = 4
3      →   teamB = [3, 1, 7, 8]
1
7
8
```

  Sample Output 1

```
1
0
3
```

4

Explanation 1

Given values are $n = 5$, $teamA = [2, 10, 5, 4, 8]$, $m = 4$, and $teamB = [3, 1, 7, 8]$.

1. For $teamB[0] = 3$, we have 1 element in $teamA$ ($teamA[0] = 2$) that is $\leq teamB[0]$.
2. For $teamB[1] = 1$, there are 0 elements in $teamA$ that are $\leq teamB[1]$.
3. For $teamB[2] = 7$, we have 3 elements in $teamA$ ($teamA[0] = 2$, $teamA[2] = 5$, and $teamA[3] = 4$) that are $\leq teamB[2]$.
4. For $teamB[3] = 8$, we have 4 elements in $teamA$ ($teamA[0] = 2$, $teamA[2] = 5$, $teamA[3] = 4$, and $teamA[4] = 8$) that are $\leq teamB[3]$.

Thus, the function returns the array $[1, 0, 3, 4]$ as the answer.

2: REST API: Top Articles

Query a REST API to get a list of articles. Given an integer, *limit*, return the top *limit* article names ordered decreasing by comment count, then decreasing alphabetically for those that have the same comment counts.

To access the collection of comments, make an HTTP GET request to:

http://hypnocore.api.hypnobox.com.br/teste/api/articles?page=<pageNumber>

where <pageNumber> is an integer where *1 <= pageNumber <= total_pages*. *total_pages* is one of the fields in the JSON data.

The response is a JSON object with the following 5 fields:

- `page`: The current page of the results
- `per_page`: The maximum number of records returned per page.
- `total`: The total number of records on all pages of the result.
- `total_pages`: The total number of pages with results.
- `data`: An array of objects containing records returned on the requested page

Each record in *data* has the following schema.

- `title`: the title of the article, may be null
- `url`: the URL of the article
- `author`: the username of the author of the article
- `num_comments`: the number of comments the article has, may be null (no comments)
- `story_id`: identifier of the story related to the article, may be null
- `story_title`: the title of the story related to the article, may be null
- `story_url`: the URL of the story related to the article, may be null
- `parent_id`: identifier of the parent of the article, may be null
- `created_at`: the date and time the record was created

First get the article name.

- If the *title* field is not null, use *title*.
- Otherwise, if the *story_title* field is not null, use *story_title*.
- If both fields are null, ignore the article.

Sort the titles decreasing by comment count, then decreasing alphabetically by article name if there is a tie in comments count. Return a list of the top *limit* names.

Function Description

Complete the function *topArticles* in the editor below.

topArticles has the following parameter(s):

int limit: the number of articles to return

Returns

string[k]: the names of articles

- **Input Format For Custom Testing**

In the first line, there is an integer *limit*.

- **Sample Case 0**