**Question1**

In this challenge, the REST API contains information about a collection of users and articles they created. Given the threshold value, the goal is to use the API to get the list of most active authors. Specifically, the list of usernames of users with submission count strictly greater than the given threshold. The list of usernames must be returned in the order the users appear in the results.

To access the collection of users perform HTTP GET request to:

https://jsonmock.hackerrank.com/api/article\_users?page=<pageNumber>

where <pageNumber> is an integer denoting the page of the results to return.

For example, GET request to:

https://jsonmock.hackerrank.com/api/article\_users/search?page=2

will return the second page of the collection of users. Pages are numbered from 1, so in order to access the first page, you need to ask for page number 1.

The response to such request is a JSON with the following 5 fields:

* page: The current page of the results
* per\_page: The maximum number of users returned per page.
* total: The total number of users on all pages of the result.
* total\_pages: The total number of pages with results.
* data: An array of objects containing users returned on the requested page

Each user record has the following schema:

* id: unique ID of the user
* username: the username of the user
* about: the about information of the user
* submitted: total number of articles submitted by the user
* updated\_at: the date and time of the last update to this record
* submission\_count: the number of submitted articles that are approved
* comment\_count: the total number of comments the user made
* created\_at: the date and time when the record was created

**Function Description**

Complete the function *getUsernames* in the editor below.

*getUsernames* has the following parameter(s):

*threshold:* integer denoting the threshold value for the number of submission count

The function must return an array of strings denoting the usernames of the users whose submission count is strictly greater than the given threshold. The usernames in the array must be ordered in the order they appear in the API response.

Input Format For Custom TestingSample Case 0

**Sample Input For Custom Testing**

10

**Sample Output**

epaga

panny

olalonde

WisNorCan

dmmalam

replicatorblog

vladikoff

mpweiher

coloneltcb

guelo

**Explanation**

The threshold value is 10, so the result must contain usernames of users with the submission count value greater than 10. There are 10 such users and their usernames in the order they appear in the API response are: panny, olalonde, WisNorCan, dmmalam, replicatorblog, vladikoff, mpweiher, coloneltcb, guelo.

**Question2**

In a warehouse, a manager would like to sort the items in the stock. Given an array of *n* item values, sort the array in ascending order, first by the number of items with a certain value,  then by the values themselves.

**Example**

*n = 6*

*items = [4, 5, 6, 5, 4, 3]*

* There are 2 values that occur twice*: [4, 4, 5, 5].*
* There are 2 values that occur once*: [3, 6].*
* The array of items sorted by quantity and then by value in ascending order is*[3, 6, 4,  4, 5, 5]*

**Function Description**

Complete the function *itemsSort* in the editor below.

itemsSort has the following parameter(s):

*int items[n]:*  an array of integers to sort

Returns:

*int[n]:* An array of integers sorted as described.

**Constraints**

* *1 ≤ n ≤ 2 × 105*
* *1 ≤ items[i] ≤ 106*

Input Format for Custom TestingSample Case 0

**Sample Input 0**

STDIN   Function

-----   --------

5        →   items[] size n = 5

3     →   items = [3, 1, 2, 2, 4]

1

2

2

4

**Sample Output 0**

1

3

4

2

2

**Explanation**

* There is a quantity of 2 for the item  2 *: [2, 2]*
* There is a quantity of 1 for the items  1, 3 and 4 *: [1],  [3], [4]*
* The array of items sorted by quantity and then by value in ascending order is*[ 1, 3, 4, 2, 2]*

Sample Case 1

**Sample Input 1**

STDIN   Function

-----   --------

10    →   items[] size n = 10

8     →   items = [8, 5, 5, 5, 5, 1, 1, 1, 4, 4]

5

5

5

5

1

1

1

4

4

**Sample Output 1**

8

4

4

1

1

1

5

5

5

5

**Explanation**

* There is a quantity of 4 for the item  5 *:[5, 5, 5, 5]*
* There is a quantity of 3 for the item  1 *:[1, 1, 1]*
* There is a quantity of  2 for the item  4 *:[4, 4]*
* There is a quantity of 1  for the items 8 *: [8]*
* The array of items sorted by quantity and then by value in ascending order is *[ 8, 4, 4, 1, 1, 1, 5, 5, 5, 5]*

**Question 3**

This challenge implements levels of friendship in Java classes.

***Acquaintance*: A person one knows slightly, but who is not a close friend.  
*Friend*: A person with whom one has a close bond.  
*Best Friend*: A person's closest friend.  
  
As the levels of friendship increase, you get to know more about the person.**

**On the basis of knowledge you have about a person: Best Friend > Friend > Acquaintance**

**Implement these levels in terms of 3 Java classes :**

**Class *Acquaintance***

* **Has an attribute: “name” (variable of type String)**
* **Constructor: Acquaintance(String name)**
* **Has a method public void getStatus which prints “[name] is just an acquaintance.\n”**

**Class *Friend***

* **Class Friend inherits class Acquaintance**
* **Constructor: Friend(String name, String homeTown)**
* **Has attribute “homeTown” (variable of type String)**
* **Has a method public void getStatus which prints “[name] is a friend and he is from [homeTown].\n”**

**Class *BestFriend***

* **Class *BestFriend* inherits class *Friend***
* **Constructor: *BestFriend*(String *name*, String *homeTown*, String *favoriteSong*)**
* **Has attribute “*favoriteSong*” (variable of type String)**
* **Has a method public void getStatus which prints *“[name]* is my best friend. He is from *[homeTown]* and his favorite song is *[favoriteSong]*.**"

Note: You do not have to worry about input handling, code stub does that

Input Format For Custom Testing

The first line contains an integer, *n*, denoting the number of friends  
Each line *i* of the *n* subsequent lines (where *0 < i < n*) contains data for a friend in the format: [**Acquaintance|Friend|BestFriend] [FriendName] {HomeTown} {FavouriteSong}**

Sample Case 0

**Sample Input**

4

Acquaintance Jaden

Friend Jake Florida

BestFriend Ryan Utah Dangerous

Friend David Texas

**Sample Output**

Jaden is just an acquaintance.

Jake is a friend and he is from Florida.

Ryan is my best friend. He is from Utah and his favorite song is Dangerous.

David is a friend and he is from Texas.

Sample Case 1

**Sample Input**

5

Acquaintance Roger

BestFriend Carson Boston Believer

Friend Oren Atlanta

BestFriend Ramon Miami Radioactive

Friend Tyson Denver

**Sample Output**

Roger is just an acquaintance.

Carson is my best friend. He is from Boston and his favorite song is Believer.

Oren is a friend and he is from Atlanta.

Ramon is my best friend. He is from Miami and his favorite song is Radioactive.

Tyson is a friend and he is from Denver.