Challenge 11 - Pheasant

« Prev Next »

There has been an outage on our event feed server, so all the generation code and the main db have been lost.

Fortunately, we have some backups of certain tables, but they are not complete. We've managed to recover all the user feed data, which is in the following format:

```
user_id timestamp_newest event_id user_id timestam-
p_newer event_id … user_id timestamp_oldest event_id
```

This data is encrypted with AES in ECB mode with a 32-byte key, with only uppercase and lowercase Latin alphabet characters.

Unfortunately, the key field of the backup table was defined with a 29 character length, so the last three digits of each key have been truncated and cannot be recovered.

We also have the last event time data for each user, which is just the timestamp_newest, and it's unencrypted.

We have not been able to recover our old generation algorithm, so we are asking you to write a new one. As you may know, our event feed is generated by ordering all friend events according to their timestamp in descending order, so you must generate a feed just like the one for Twitter. Customers don't need your entire feed, so there is an events number limit as an input, along with your friends list.

The feeds and the last times are available in the following file: feeds.tar.gz

Input

The input contains a list of cases to solve, one per line.
Each line (or case) consists in the number of events we need **N** ($1 \leq \mathbf{N} \leq 300$), followed by the list of user id's of your friends and the keys of each of them with at most 3000 users in that list.

Output

You have to print the solution for each case in a separated line.
For each case, the output should be the list of the event ids requested, separated by spaces. If users don't have enough events to reach the limit, simply return the events they have. However, keep in mind that they must be sorted by timestamp in descending order.

Input format

```
events_number; user_1_id,user_1_key;
user_2_id,user_2_key; …
```

Output format

```
event_id_1 event_id_2 event_id_3…
```

Sample input

```
7; 17023,hNMqffpeMSqUqNfbvSDImDRQmtSbU;
57970,ZWIiWjrkhhlEJcSnOuCAnXQqexSxC;
88916,aMLtuoZOwrydHyXBJCexjkzeBGKvF;
94293,CItCenDzXLglVHJqZSrkdGWatYnrg;
21533,mtGlWjHkfkthdDXzRCfsjtVssCNDO;
89112,fXMAPXYluCCUHpefOrNTvIOUtyrJQ;
97879,rfUvNLbFhqeqglWWYztSgRXVoHlcn
20; 24969,MjFhBiAdhApDjxPUPTAYEmkipNdjl;
40446,GrkLwoykYYLDKhLJmuDxEIAOOpQwa;
52107,CbsYzqgwTHIfMlPKKfeekviPYAmRE;
51797,PzXWkkJZrbyiBlACjaVlRNJtCStQZ;
18991,yzcVwTbZhPZNKEbdEusitxTfiGPIg;
53847,bcotHVZiQBNRhmyEbpepjdPuYlkYq;
93538,hlIuekdFLMwOfXqoXNZhUEGQhcfrs;
101158,PVjneTkBAPTuvePutVpUcmsjsyVyV;
78771,tHCREWafBRhzTXbhNleVkeSgWLvYW;
55239,gTPVujaEJFIEtgVbfXOvLsLtKpVRw;
```

```
86644,CiugbjSOnyPmMoXAHAXEQCvweErbJ;
24416,ZlsDCLXYUaDTpnybiJOAHOthWYdrL;
94329,IrMYwnrdwqqCQuAnSHQHhHwXGNzHM;
83021,ldSSaLpogUfneQXtxdkXtWWbHIPjv;
64346,zskHxtWYdlxKagqxZOYOYlylGATCH;
56253,RwJDSFjUCadVLnZpALcJLKZZblWke;
74977,ccuOqhjoCDSUKamYEsKuXdNqqlhjk;
103658,EKjXiUZcytVrLAqDmIjZsqhjMMgLG;
43021,UYvTkcqhmZSHaknsQpNTgkggscJtX;
101157,ybQJOoPdPaophiGhTJpqfkdHSzrNV;
77118,dWwpxTxuomGKVhRnKfVgOucoLOPwq;
31585,TZrYqgkEqaKjQntAOGPMOqWrGrzBX;
101618,mzKJmsDhrgMUZaSloGePrUmljnRzu;
102425,YrftsSyQlbBvQihtPZjEllmQIOMaw;
66314,oRmtDeavEYfoqHdViFMPKsppzfdYI;
95219,ompwtQaKZCfRZWAmoNMImWMelEzCn;
93822,xWzvFiDomRbQbzuqHcpLBAlKlKyoY;
28868,vPJAdmuPeQQoPTmTRyFuhefsmHbTM;
80171,QlJzvVAwBlCrYGhBYMFjfUtPcrmmB;
42535,OrqGLsKETQPCSzPbRqoQKouWwsyug
5; 63321,YwsSygLoCYQvyrnMrqmZZigmwMdyq;
79648,CsBZurtvHMsdqIWcsUsgEldfbhYSR;
50262,dsJIQeLJNhVeeBJEwIfGbpERHSelS;
89350,fRaJaPBUiNkvSljaloVDkpytCsFYS;
43756,IajoLdozmQVcVxCReEQlFGuPbtpkj
```

Sample output

```
2379660 2356841 2356830 2323784 2293275 2268941 2240228
2524521 2524518 2524517 2524514 2524509 2524497 2517572
2517558 2517551 2510502 2510492 2510491 2510486 2510479
2510478 2510473 2510469 2510458 2503510 2503509
2519439 2492300 2487373 2472350 2424300
```

Submit & test your code

To test and submit code we provide a set of tools to help you. Download contest tools if you haven't already done that. You will then be able to test your solution to this challenge with the challenge tokens.

```
Challenge tokens: CHALLENGE_11, CHALLENGE_SUBMIT_11
```

To test your program

```
./test_challenge CHALLENGE_11 path/program
```

A nice output will tell you if your program got the right solution or not. You can try as many times as you need.

To test your program against the input provided in the submit phase

```
./test_challenge CHALLENGE_SUBMIT_11 path/program
```

During the submit phase, in some problems, we might give your program harder inputs. As with the test token, a nice output will tell you if your program got the right solution or not. You can try as many times as you need.

In the actual contest you first need to solve the test phase before submitting the code, you must provide the source code used to solve the challenge and you can only submit once (once your solution is submitted you won't be able to amend it to fix issues or make it faster).

If you have any doubts, please check the info section.

« Prev Next »