ACM/ICPC Asia Kabul
National Programming Contest

Problems

Organized by ACM/ICPC Kabul Team
Sept, 2019

1. FRSUG

Ahmad is a software engineer in Xyz company. He is working on an algorithm for the friend suggestion functionality. The algorithm needs the number of people and how they are close to each other (100 means don't know each other, and 1 means they are close friends, or family member) as input. You need to write a program to help him.

Input:

First line contains the number of people as N. (0 <= N <= 500)
Each next N lines contains three integers A, B (represent is the ID for two people), and W (how close A and B are). (0<=A, B<=500, 1<=W<=100)
Next line contains a person's ID as U, which we want to suggest him friends.
Next line contains the number of queries we want to perform as Q. (1<=Q<=500)
Each next Q lines contains IDs for the target person as V. (0<= U, V <=500)

Output:

For each query, print how close the person is to the ID specified, or '----' if they have no connection at all.

Example:

Input:

```
7
0 1 4
0 3 8
1 4 1
1 2 2
4 2 3
2 5 3
3 4 2
0
4
1
4
5
7
```

Output:

```
4
5
9
----
```

Description:

Query #1: 0->1 =>	closeness = 4
Query #2: 0->4 => 0->1->4	closeness => 4+1 = 5
Query #3: 0->5 => 0->1->2->5	closeness => 4+2+3 = 9
Query #4: 0->7 => ----	no connection at all

2. DISTCODE

Ahmad recently learned about country codes - two letter strings, denoting countries. For example, BY stands for Belarus and IN stands for India. Mesmerized by this new discovery, Ahmad now looks for country codes everywhere!

Ahmad has recently found a string S consisting of uppercase Latin letters. He wants to find the number of different country codes that appear in S as contiguous substrings. For the purpose of this problem, consider that every 2-letter uppercase string is a valid country code.

Input:

The first line of input contains an integer T denoting the number of test cases. $1 \leq T \leq 100$

Each next T lines contains a string S, consisting of uppercase English letters. $0 < \text{len}(S) < 10000$

Output:

For each test case, print a single line containing the number of different country codes appearing in the given string.

Example:

Input:

```
2
INBY
BYBY
```

Output:

```
3
2
```

3. UPDATES

A software company wants to build a very important module for their desktop software - The update checker, which is responsible for checking if there are new updates on the server. A module needs to be updated if the same module has a higher version number in the server. So, It simply compares the identifier of modules installed in the desktop software with the ones in the server to find the list of updates that need to be installed. Write a program that gets the identifier lists of installed modules and the ones on the server as input, then prints the list of updates that need to be installed.

Input:

First line contains `S`:

- `S` is the list of modules identifier separated by commas on the server.

  `0 < size(S) < 10000`

Second line contains `I`:

- `I` is the list of installed modules identifier separated by commas.

  `0 < size(I) < 500`

Module identifier format: `[module-name].v[module-version]`

- examples:
  - `the.v1.version-of_xyz_MODULE.v0.0.0.20191010081012`
  - `the.v1.version-of_xyz_MODULE.v0.1`
  - `x.v9`
  - `x.v99.123.23423.20180223120001`
- `0 < len(module-name) < 100`, and contains (`a-z,A-Z,0-9,.,_,-`) characters
- [`.v`] separate the module name and module version
- Module-version format: `major.minor.macro.build-number(datetime)`
  - `major < 100`
  - `minor < 1000` and is optional
  - `macro < 100000` and is optional
  - `build-number` format is `YYYYMMddHHmmss` and is optional

Output:

Print the list of modules updates that need to be installed on desktop (separated by commas and sorted by their module name in ascending order).

Note:

- Modules can have multiple updates. eg: x.v1 => x.v1.0.1 => x.v1.1.1

Check the example in next page.

Example:

Input:

```
a.v1,b.v2.0.1,c.v1.1,d.v0.0,e.v5.0.9,f.v0.0.0.20191010101010
a.v1,c.v1.1.0.20191010101010,e.v5.1,f.v0.0.0.20191010101011
```

Output:

```
c.v1.1.0.20191010101010,e.v5.1,f.v0.0.0.20191010101011
```

Description:

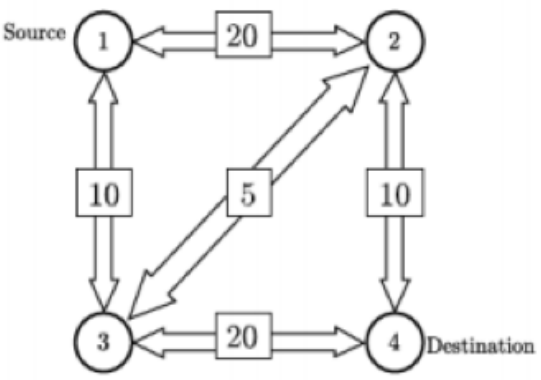
If we create a table based on the given input, and compare the version numbers. Then we only need to update the three colored module in the desktop software. We also have two modules in server that we don't have it in the desktop, so we ignore them.

Server Modules	Relation	Installed Modules
a.v1	==	a.v1
b.v2.0.1	x	
c.v1.1	<	c.v1.1.0.20191010101010
d.v0.0	x	
e.v5.0.9	<	e.v5.1
f.v0.0.0.20191010101010	<	f.v0.0.0.20191010101011

4. INBDT

In the Internet, machines (nodes) are richly interconnected. Many paths may exist between a given pair of nodes. The total message-carrying capacity (bandwidth) between two given nodes is the maximal amount of data per unit time that can be transmitted from one node to the other.

Using a technique called packet switching, this data can be transmitted along several paths at the same time. For example, the figure shows a network with four nodes (shown as circles), with a total of five connections among them. Every connection is labeled with a bandwidth that represents its data carrying capacity per unit time. In our example, the bandwidth between node 1 and node 4 is 25, which might be thought of as the sum of the bandwidths 10 along the path 1-2-4, 10 along the path 1-3-4, and 5 along the path 1-2-3-4. No other combination of paths between nodes 1 and 4 provides a larger bandwidth.



You must write a program that computes the bandwidth between two given nodes in a network, given the individual bandwidths of all the connections in the network. In this problem, assume that the bandwidth of a connection is always the same in both directions (which is not necessarily true in the real world)

Input:

The first line contains the number of test cases as T. (0 < T < 10)

- First line contains the number of nodes (numbered from 1 to N) in the network as N (2 ≤ N ≤ 100).
- The next line contains three numbers: S, and T are the source and destination nodes, and C is the total number of connections in the network. (0 < C < 100)
- Each next C lines contain three integers: the first two represent the connected nodes, and the third one is the connection bandwidth as B. (0 <= B <= 1000)

Output:

For each test case print the total bandwidth between the S and T.

Note:

- There might be more than one connection between a pair of nodes, but a node cannot be connected to itself
- All connections are bi-directional, i.e. data can be transmitted in both directions along a connection, but the sum of the amount of data transmitted in both directions must be less than the bandwidth.
- A line containing the number '0' follows the last network description, and terminates the input.

Check the example in next page.

Example:

Input:

```
1
4
1 4 5
1 2 20
1 3 10
2 3 5
2 4 10
3 4 20
```

Output:

```
25
```

5. BTAR

A sequence of integers is beautiful if each element of this sequence is divisible by 4.

You are given a sequence a_1, a_2, ..., a_n. In one step, you may choose any two elements of this sequence, remove them from the sequence and append their sum to the sequence. Compute the minimum number of steps necessary to make the given sequence beautiful.

Input:

The first line of the input contains a single integer T denoting the number of test cases. $1 \le T \le 10^5$

- The first line of each test case contains a single integer n. $1 \le n \le 10^5$
- The second line contains n space-separated integers a_1, a_2, ..., a_n.

  $1 \le$ sum of n over all test cases $\le 10^6$

  $0 \le a_i \le 10^9$

Output:

For each test case, print a single line containing one number — the minimum number of steps necessary to make the given sequence beautiful. If it's impossible to make the sequence beautiful, print -1 instead.

Example:

Input:

```
1
7
1 2 3 1 2 3 8
```

Output:

```
3
```

6. AVGL

The Xyz University in Afghanistan wants to create a program to help them with their reporting. They want to create a report which is a list of students sorted descending by their final average mark (top student first). It is important to note that universities in Afghanistan are using a custom form of Credit System (how students select courses, and they are counted toward their degrees) in their universities:

- Each semester, students shall follow the same study plan and study specific courses (each with a specific number of credit).
- Student will fail the course, If s/he gets lower than 50% mark for the course.
- Student will fail the semester, if s/he fails more than 50% of the credits.

Input:

First line contains the number of test cases T: `0 < T < 100`

- Next line contains the number of courses (C) for current test case:
  `0 < C < 10`
- Next line contains the courses name (CN) and its associated credit (R) (separated by whitespace and commas). `0 < R < 5, 0 < len(CN) < 100`
- Next line contains the number of students (S): `0 < S < 100`
- Next S lines contains name (N), and marks (M) for each student (in the same order of course-credit list) separated by commas. `0 < len(N) < 100, 0 <= M (Double) <= 100`

Output:

Print the list of students sorted in descending - the top student in top using the following format:

- std-name,mark-avg(up to 2 decimal digits),num-of-failed-courses,failed-semester-or-no?
  `Eg: S1,89.23,0,false`
- Students list must be sorted by their mark-average, BUT remember:
  - Students with no failed course comes first,
  - Students with fewer number of failed credits comes next.
  - If mark-average, number of failed courses and credits are the same, sort by name in ascending.

Separate each test case output with an empty line (no empty line after the last testcase result).

Check the example in next page.

Example:

Input:

```
1
2
A 4,B 2
5
S1,80,56
S5,100,0
S3,100,100
S4,56,89
S2,49,92
```

Output:

```
S3,100,0,false
S1,72,0,false
S4,67,0,false
S5,66.66,2,false
S2,63.33,4,true
```

Description:

Below is a table that shows the calculated value for the student marks:

	A	cr	B	cr	total	avg
S3	100	4	100	2	600	100
S1	80	4	56	2	432	72
S4	56	4	89	2	402	67
S5	100	4	0	2	400	66.66
S2	49	4	92	2	380	63.33

S3 has the highest average, with S1, and S4 coming next. Both S5, and S2 failed some credits; but because S5 has failed only in 2 credits (s/he failed the course but because it is less than 50% of total credits - 6, s/he is not failing the semester) comes before S2 which failed in 4 credits (and the semester).

7. NUKES

There are K nuclear reactor chambers labelled from 0 to K-1. Particles are bombarded onto chamber 0. The particles keep collecting in the chamber 0. However if at any time, there are more than N particles in a chamber, a reaction will cause 1 particle to move to the immediate next chamber (if current chamber is 0, then to chamber number 1), and all the particles in the current chamber will be destroyed and the same continues till no chamber has number of particles greater than N. Given K,N and the total number of particles bombarded (A), find the final distribution of particles in the K chambers. Particles are bombarded one at a time. After one particle is bombarded, the set of reactions, as described, take place. After all reactions are over, the next particle is bombarded. If a particle is going out from the last chamber, it has nowhere to go and is lost.

Input:

The first line contains the number of test cases. 0 < T < 1000000

- Each next T lines contains three numbers A,N and K separated by spaces.
    - 0 < A < 1000000000
    - 0 < N < 100
    - 0 < K < 100
- All chambers start off with zero particles initially.

Output:

For each test case print three numbers separated by whitespace:

- The first number is the number of particles in chamber 0,
- the second number is the number of particles in chamber 1
- and so on.

Example:

Input:

```
1
3 1 3
```

Output:

```
1 1 0
```

8. SHSUB

Given a set of `N` names, find the smallest string that contains each name in the given set as substring. We may assume that no name in set is substring of another name.

Input:

First line contains the number of test cases `T`. Next `T` lines contains commas separated strings.

Output:

Print the smallest string that contains each name in the given set as substring for each test case.

Example:

Input:

```
geeks,quiz,for
catg,ctaagt,gcta,ttca,atgcatc
```

Output:

```
geeksquizfor
gctaagttcatgcatc
```

9. RICHSTR

A string with length L is called rich if $L \geq 3$ and there is a character which occurs in this string strictly more than L/2 times.

You are given a string S and you should answer Q queries on this string. In each query, you are given a substring $S_L, S_{L+1}, \ldots, S_R$. Consider all substrings of this substring. You have to determine whether at least one of them is rich.

Input:

The first line of the input contains a single integer T denoting the number of test cases.

$1 \leq T \leq 10$

- The first line of each test case contains two space-separated integers N and Q.
  $1 \leq N, Q \leq 10^5$
- The second line contains a single string S (contains only lowercase English letters) with length N.
- Each of the next Q lines contains two space-separated integers L and R describing a query.
  - $1 \leq L \leq R \leq N$

Output:

For each query, print a single line containing the string "YES" if the given substring contains a rich substring or "NO" if it does not contain any rich substring.

Example:

Input:

```
1
10 2
helloworld
1 3
1 10
```

Output:

```
NO
YES
```

10. T2B

Consider an N*N matrix, suppose each cell in the matrix has a value assigned. We can go from each cell in row i to a diagonally higher cell in row i+1 only [i.e from cell(i, j) to cell(i+1, j-1) and cell(i+1, j+1) only]. Find the sum of values of the path from the top row to the bottom row following the aforementioned condition such that the maximum sum is obtained.

Input:

First line contains number of matrix rows as N (matrix is squareness). 0 < N < 1000

Each next N lines contain N numbers, which are the cells' values of the Nth row.

Output:

Print the maximum sum of values of the path from the top row to the bottom row following the aforementioned condition.

Example:

Input:

```
4
5 6 1 7
-2 10 8 -1
3 -7 -9 11
12 -4 2 6
```

Output:

```
28
```

Description:

The highlighted numbers from top to bottom gives the required maximum sum path.

```
      {5, 6, 1, 7},
      {-2, 10, 8, -1},
      {3, -7, -9, 11},
      {12, -4, 2, 6}

7 + 8 + 11 + 2 = 28
```