Table of Contents

1. BANDWIDTH

A tiny network utility reports the current internet download/upload speed. It simply shows the total number of bytes transferred per second. You should write the core function of this utility. Given a series of download/upload transferred bytes, calculate and print download/upload speed per second.

Input

- First line of the input contains number of test cases (***0 < number of test case < 1000***).
- For each test case the first line contains the number of download/upload bandwidth usage that comes after (***number of lines < 100000***).
- The download/upload bandwidth usage format is as follow:
    - *time-in-millisecond*[whitespace]*bytes-downloaded*[whitespace]*bytes-uploaded*
    - ***0 <= time-in-millisecond (from the moment app is started) <= 2^64***
    - ***0 <= bytes downloaded/uploaded <= 10485760 (10MB)***

Output

- Use the phrase '---' to separate the outputs for each test case. The phrase doesn't come before the first and after the last test case's result.
- Print the speed in humanized format using B, KB, MB, GB, TB (Kilo = 1024, B = Byte), with at most two digits of decimal (rounded to the nearest value) in following pair format:
    - *humanized-download-speed-per-second/humanized-upload-speed-per-second*

Example

Input

2
7
120 1 0
600 200 30
999 120 80
2000 1 1
2230 800 10
2700 38 9999
2980 10 10
1
679 1572864 51200

Output

321B/110B
0B/0B
849B/9.79KB

1.5MB/50KB

2. STRINTERPOLATION

Khaled is working on an open source library. A very interesting feature of the library is the support of string interpolation. String Interpolation is the process of replacing parameters in a string. Khaled wants the library to support recursive interpolation too. Help Khaled on writing this piece of the library.

Input

- First line of the input contains the number of test cases (***0 < number of test case < 1000***).
- For each test case, the first few lines contain the parameters mapping (***number of parameters mapping's lines < 1000***). The parameter mapping format is as follow:
    - *key=value*
    - ***0 < key is a text < 50***
    - the key doesn't contain '=' symbol
    - ***0 <= value is a text < 1000***
- Then it comes the actual string that needs to be interpolated. The string starts and ends with a line containing only the '---' phrase (***0 <= string is multiline <= 10240***).
    - Parameters are marked using '***${param}***' format.
    - The library doesn't support escaping the '***${param}***' itself.
    - The input string doesn't contain partial form of '***${param}'***.
    - The input string interpolation will NOT result to an infinite recursive call

Output

- For each test case print the interpolated input string.
    - If param is not defined in the map, use the param's key as the value.

Example

Input

```
2
a=Khaled
b=${a} is ${c} year's old
c=25
---
Hi, ${b}
---
---
---
```

Output

```
Hi, Khaled is 25 year's old

[an empty line for second test case]
```

3. BARRELS

There is a farm with many cows. Each cow produces a few liters of milk per shift in a day. The farmer has barrels of specific volume. Given the liters of milk every cow produces per shift and per day, find out how many barrels does the farmer need.

Input

- The first line of input contains the number of test cases (***0 < number of test case < 1000***).
- Test case input data are separated by an empty line.
- The first line of each test case specifies the barrel volume (***0 < volume in liter < 100)***.
- Each next line in a test case contains the number of liters a cow produced milk per shift in a day separated by a space.
    - ***0 < number of shifts <= 5***
    - ***0 <= number of liters per shift <= 100***

Output

- For each test case, print the number of barrels the farmer needs.

Example

Input

2
25
1 2 5 20
32 40
20
0

10
100
[empty line]

Output

5
10

4. SCHEDULER

Zahra is a university student interested in working on a simple job scheduler. The scheduler executes the submitted tasks based on very simple rules: 1) tasks are executed in order they are submitted, 2) scheduler executes the current task only for specific amount of time, 3) if the current task is not finished, it is re-submitted to the scheduler to be executed later. Zahra wants to test her implementation and see if it works correctly. Given a list of tasks along with how long it will take to complete, find out the order they complete their execution. So, Zahra can compare your output with her implementation's result.

Input

- The first line of input contains the number of test cases (***0 < number of test case < 1000***).
- The first line of each test case contains the specific duration scheduler spent on each execution, and the number of tasks that will are submitted initially separated by a whitespace (***0 < per-execution duration in ms < 10, and 0 < number of initial submitted tasks < 1000***).
- For each test case, each next line contains the task name and how long it will take to complete (***0 < task name is unique alphanumeric < 32, and 0 < duration in ms < 1000***)

Output

- For each test case print a line which contains task names in order they completed separated by a whitespace.

Example

Input

```
2
1 2
task1 10
task2 2
2 1
task1 100
```

Output

```
task2 task1
task1
```

5. PROJMNG

The company X is working on a new product – an online project management tool. There is a very essential feature they have to implement – finding the project duration. Given a list of tasks along with their dependent tasks and duration, they must calculate the project's duration.

Input

- The first line of input contains the number of test cases (***0 < number of test case < 1000***).
- Each test case data is separated by an empty line.
- Each line in a test case contain the task name, duration in minutes, and dependent tasks (all dependencies are valid task names) in following format:
    - *task-name[whitspace]duration-in-mins[whitspace]dependent-taskname,…*
    - ***0 < task-name is unique alphanumeric text < 32***
    - ***0 < duration in minutes < 10000***
    - ***0 <= number of dependent tasks < 100***

Output

- For each test case print the project duration in minutes

Example

Input

2
task1 10 task2 task3
task2 20
task3 32 task2

task1 100
task2 90

Output

62
100

6. SECRETMSG

Jafar and Ebrahim developed a simple chat app that works in a LAN network. It is meant to be a fun project, and not for commercial use. They want to add a simple encryption technique to secure the messages. The technique is based on a simple keypad number to number/alphabet mapping.

Input

- The first line of input contains the number of test cases (***0 < number of test case < 1000***).
- Each test case is a single line which contains sequences of digits (0-9) (***0 < number of digit-sequence < 1000***) separated by whitespaces. Note that all sequences will map to a valid digit or character.
- Use the image on the right as the mapping reference: e.g.:
    - 1 => 1
    - 2 => 2, 22 => a, 222 => b, 2222 => c
    - 3 => 3, 33 => d, 333 => e, 3333 => f
    - …

Output

- For each test case print the decrypted message.

Example

Input

2
444 4444
44 6666 6666 33 00 5555 888 2222 555

Output

hi
good luck

7. NIDORDER

The NSIA office wants to add an amazing feature to the NID (national id) online portal. Currently they call each person/family to tell them when to visit the NID office to process their NID application. But this is super slow, and not efficient. They want to estimate the date and time when the applicant should visit the office. Although this feature should be very advanced, for now, they want to implement it as simple as the following: Given a list of steps each person should follow along with the duration it takes, and the fact that we should process the applications in order they are submitted. We should calculate how long processing a person/family application will take and assign them an estimated date and time to visit the NID office.

Input

- The first line of input contains the number of test cases (***0 < number of test case < 1000***).
- For each test case the first few lines contain the steps along with its estimated needed duration (***0 < number of steps < 10***) in following format:
    - *step-name=duration-in-minute*
    - ***0 < step-name is alphanumeric < 20***
    - ***0 < duration is in minute <= 480***
- Then it comes the list of applicants count per family (***0 < number of applicants per family < 20***) in following format:
    - *number-of-applicants-in-a-family*
- Input ends with an empty line

Output

- For each test case print the estimated date and time (starting from [***1 8:0***] in following format:
    - *day hour:min*
    - *e.g.: 2 10:50 <= at 10:50 of second day*
    - ***the day starts from 1***
    - ***the hour and mins start from 08:00 (morning) and ends to 16:00 (afternoon)***
    - ***an application can be processed in span of multiple day, but each step must start and end within the specified time rang***
    - ***Imagine there is only one NID employee processing applications serially, starting exactly from 08:00 to 16:00 with no pause***
    -

check the next page for an example…

Example

Input

2
reviewing-application=5
confirming-identity=4
step-x=10
step-y=20
1
9
2
the-only-step=60
3
10
5
[empty line]

Output

1 8:0
1 8:39
1 14:30
1 8:0
1 11:0
2 13:0

day	time	note
day 1	8:00	first family
	8:39	second family
	9:18	
	9:57	
	10:36	
	11:15	
	11:54	
	12:33	
	13:12	
	13:51	
	14:30	third family
	15:09	

test case 01

day	time	note
day 1	8:00	first family
	9:00	
	10:00	
	11:00	second family
	12:00	
	13:00	
	14:00	
	15:00	
day 2	8:00	
	9:00	
	10:00	
	11:00	
	12:00	
	13:00	third family
	14:00	
	15:00	
	16:00	
	17:00	

test case 02

8. ELECENG

Ahmad started working as an electrical engineer at DABS (Da Afghanistan Breshna Sherkat). He is assigned to lead the Transformers Installations Project in multiple provinces. Every province has 'V' districts, and there are only 'T' transformers available for installation in that province. He wants to design the electricity grid to keep the distance between transformers and districts at a minimum.

Input

- The first line of input contains the number of test cases (***0 < number of test case < 10***).
- For each test case, the first line contain contains the T and V in 'T V' format.
    - Each next V lines contain the distance between districts (adjacency matrix)
      d1,d2,d3,…,dv
      d1,d2,d3,…,dv
      ...
      d1,d2,d3,...,dv
    - ***0 < T, V < 10***

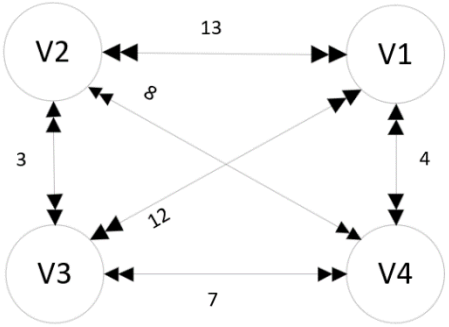
Output

- For each test case, in the first line output the max distance between a district and the transformer, and in the next line output the district's number connected to the transformer separated with commas.

Example

Input

2
2 4
0,13,12,4
13,0,3,8
12,3,0,7
4,8,7,0
2 5
0,2,6,9,8
2,0,4,10,11
6,4,0,5,12
9,10,5,0,7
8,11,12,7,0



first testcase visualization

Output

4
1,2
7
1,4

9. TOMARS

Ahmad is the last person left on earth, but he wants to get to the mars station fly to and join others on the mars. There are multiple stations along the way, on each station he tries to find the fastest car to get to the next station. On each station, there are information about the distance to the next station, and each car's speed, fuel, and fuel consumption per kilometer.

Input

- The first line of input contains the number of test cases (***0 < number of test case < 100***).
- For each test case,
    - The first line contains the number of stations along the way to mars station
    - The next line contains the Ahmad's distance to next station
    - The next line contains comma separated groups of data with 'speedN-chargeN-chargeSpentN' format
    - ***0 < speedN < 300***
    - ***0 < chargeN < 500***
    - ***0 < chargeSpentN < 500***

Output

- For each test case,
    - On the first line, output the duration (rounded to three decimal) that it takes Ahmad to get to the mars station
    - And on the next line, output the cars group number that Ahmad took on each station separated by whitespace

Example

Input

```
1
3
6
10-20-5,30-30-5
5
30-40-5,60-30-10
3
80-40-10,70-30-10
```

Output

```
0.404
2 1 1
```