Table of Content

CPU

Mahmood is a programmer in SoftAfghan company. Mahmood's team works on a national Operating System. Design and development of a scheduling algorithm for tasks distribution among existing CPUs is assigned to Mahmood. He should write a program that assigns background tasks with predetermined start and finish time to the CPUs such that the least possible CPUs are used. Then, free CPUs can be assigned to the foreground tasks. Help him in this regard.

Input

The first line indicates the number of cases. In each case, the first line is the number of tasks (T). The second and third lines are the sequences of start and finish time for each task.

$1 \leq T \leq 100$

$0 \leq$ Start Time and Finish Time ≤ 100

Output

Minimum number of CPUs that are occupied by the background tasks for cases, each one in a new line.

Sample

Input

```
2
7
1 3 2 0 5 8 11
3 4 5 7 9 10 12
5
1 3 5 6 8
5 6 8 8 9
```

Output

```
3
2
```

12T24

Ahmad has started to work as an intern at Company XYZ. As his first task, he should write a program to fix the log file entries time format. The function that receives the time string as input, and returns the formatted time string is the main part of this program. He wants you to write this function for him.

Input

First line contains a number N - the number of test cases (0 < N < 10). Each next line (which represents a test case) contains a valid time string - time format "`\d{1,2}:\d{1,2}(am|pm)?`".

Output

For each test case, print the time in 12hr format using "`\d{2}:\d{2}(am|pm)`".

Sample

Input

```
3
0:0
01:1pm
20:20
```

Output

```
12:00am
01:01pm
08:20pm
```

SUGITEMS

In online e-commerce platforms, a simple recommendation system works like this: user navigations (through items) are stored then analyzed to understand what are the next possible items that a user may navigate to. This can be simply done by finding what are the most frequently visited items after each item. Now that you know how it works, given the user's navigation history, find and recommend top items for each.

Input

First line consists of a number U - the number of user navigation histories (0 < U < 10).

Each next line contains a string showing the navigation between items (each item is represented by an English letter - case insensitive).

Output

For each item (per line - chronologically), print the recommended items in descending order (most recommended first, and chronologically if multiple items have the same recommendation level - if any) separated by commas.

Sample

Input

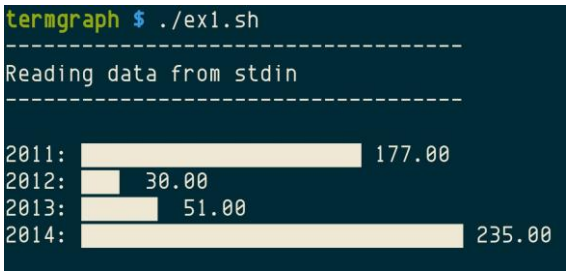
```
3
a>b>c>d
a>c>b
b>c>a
```

Output

```
a:b,c
b:c
c:a,b,d
d:
```

BARCHART

Ahmad is working on a command line financial tool. Generating a bar chart is one of the features for this tool. Ahmad wants you to develop this part since it is going to be an open source and wants you to be a part of it.



Input

First line consists of a number N - the number of test cases (0 < N < 10). Each next line (which represents a test case) starts with a number - W, the number of letter L to appear in the max column - barchart width (5 < W < 30), a letter - L, the letter to use for drawing chart's column (any letter), and X pairs of column values "ColumnName:ColumnValue", (0 < length(ColumnName) < 10, 0 < ColumnValue is a decimal number <= 999, 0 < X < 10) all separated by whitespace.

Output

For each test case, print a simple bar chart. Note that ColumnName is right aligned (length=5), ColumnValue should be formatted with 3 digits before, and 2 digits after the decimal point. Round your result when calculating column's height.

Sample

Input

```
1
10 # a:100 b:50 colC:0
```

Output

```
    a: ########## 100.00
    b: #####  50.00
 colC:    0.00
```

ELVSEQ

Jawid is an enthusiastic programmer, and he wants to revolutionize how elevators work. He wants to work on a more efficient elevator algorithm but for now he has to write the usual elevator algorithm. The elevator algorithm can be summarized as follows:

- Continue traveling in the same direction while there are remaining requests in that same direction.
- If there are no further requests in that direction, then stop and become idle, or change direction if there are requests in the opposite direction.

Give him a hand, and write this algorithm for him.

Input

First line consists of a number N - the number of test cases (0 < N < 10). Each next line (which represents a test case) contains X floor numbers F (1 < X < 10, 0 <= F <= 10) entered by the passengers (in the order they are entered), separated by whitespace. Note that the first number is the floor on which all passengers get into the elevator. The initial direction is determined by the first two floor numbers.

Output

For each test case, print the sequence of floor levels which elevator will stop (including the initial floor level).

Sample

Input

```
2
1 6 3 2 5 4
5 6 2 3 4
```

Output

```
1 2 3 4 5 6
5 6 4 3 2
```

TXTCMP

Basically a dictionary compression algorithm looks for patterns in the given data, and then substitutes those patterns with smaller replacements resulting in the final data being shorter/smaller. Great, then lets create a super simple text compression tool. It will break the given string into smaller chunks of equal length, then associate and replace each unique chunk with a shorter placeholder text.

Input

First line consists of a number N - the number of test cases ($0 < N < 10$). Each next line (which represents a test case) starts with a number - C, the chunk length ($0 < C < 10$), and the text to be compressed - T ($0 < \text{length}(T) < 1000$).

Output

For each test case, print the compressed text. Use a numeric sequence (starting from 0 up to infinity) as the chunk's placeholder.

Sample

Input

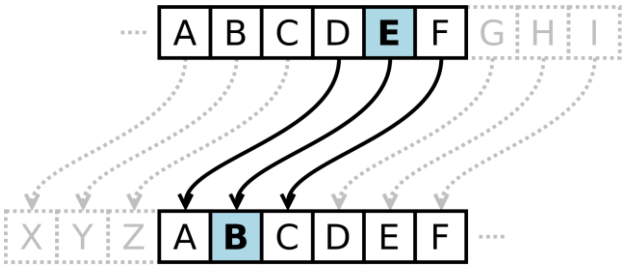
```
2
2 aabbccddeeabcde
3 abcdefabcdef
```

Output

```
12345678
1212
```

CAESAR

Cryptography is a very interesting topic. Caesar cipher is a simple cryptography algorithm that was used by and named after Julius Caesar ~100 BC. It is a simple monoalphabetic substitution algorithm in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. Implement a caesar encryption and decryption program.



source: https://en.wikipedia.org/wiki/Caesar_cipher

Input

First line consists of a number N - the number of test cases (0 < N < 10). Each next line (which represents a test case) starts with either "encrypt" or "decrypt" specifying the operation, the shift value S (-26 <= S <= 26), and the text T (0 < length(T) < 100, and T contains only English lowercase alphabets).

Output

For each test case, print the encrypted or decrypted text.

Sample

Input

```
2
encrypt 1 abc
decrypt -1 acd
```

Output

```
bcd
zbc
```

STUDY

Successful students usually follow an intelligent study plan. As part of preparing such a plan, they consider the hours they have to prepare for the exam (T), the number of references they need to review (R), how many hours is required to review each reference (C) and the value of each reference based on the number of questions that will be extracted from each reference (P).

Design a program that suggests an intelligent study plan for students so they can answer most possible exam questions based on their time for studying and the value of the references and the time it takes to review them.

Input

Input consists of a number of cases. First line shows T and R followed by R lines; each one is a couple of C[R] and P[R] for reference. You may assume that a reference's questions can be solved if the reference is reviewed completely. The input is ended by the case with T=0, R=0. This case should not be processed (T and R $\leq$ 20).

Output

For each input case, print one line of output that shows the maximum number of problems that can be solved by the student if follows the suggested study plan.

Sample

Input

```
3 2
1 5
3 8
5 3
2 5
2 5
4 9
0 0
```

Output

```
8
10
```

ROAD

Road construction engineer Mrs. Arzu is responsible for the design of roads that connect provincial centers. The project managers asked her to find a route that requires a lower budget for road construction among the routes connecting the provinces' centers. Help her to report to the project managers the amount of budget saving provided that the provincial centers are connected efficiently instead of building roads in all routes. The route list between the provincial centers and the budget required for road construction are in her hands.

Input

The first line consists of two integers that show the number of provinces (P) and routes among them (R), respectively. For each route, there is an input line that has three numbers indicating roads' source, destination and required budget (in million dollars). Note that each city has a numeric sequential code, start from 0.

0 < P <= 50

P-1 <= R <= P^2

Output

Amount of money saved.

Sample

Input

```
3 3
0 2 2
0 1 4
1 2 1
```

Output

```
4
```

OVRSPED

Given the points, and the distance between each two (route), along with the speed limit for each route, write a program to find if someone can get from point A to B without overspeeding, within the given time.

Input

First line consists of a number N - the number of test cases (0 < N < 10).
Eac test case ends by an empty line. Each test case starts with lines (0 < num of lines < 10) each defining the directional connection between two points, along with the distance and speed limit (format: `point1->point2:distance,speedlimit`) - points are represented by English letters; distance and speed limit are measured in kilometers (0 < distance < 100, 0 < speed limit < 100) and may have decimal points upto 2 digits. The remaining lines in each test case (0 < num of lines < 20), each represent a query - if one can get from one point to another within the given time without overspeeding (format: `point1->point2:time`) - time is measured in minutes (0 < time < 1000).

Output

For each test case's query, print "Yes" if it is possible to perform the trip, and "No" if not (due to non-existing point/route, or not being able to make it on-time). Separate test case outputs with an empty line.

Sample

Input

```
2
a->b:5,10
a->c:100,60
b->c:10,15
a->b:30
a->c:60

a->b:10,10
a->b:61
```

Output

```
Yes
No

Yes
```

LTRFRQ

Hussein is doing some research on the English alphabet. He is trying to understand how often each letter has appeared in a text. He wants to find their frequencies. Since he is not familiar with programming languages, he is asking for your help. Write him a program which will count the number of times each letter appears in the given text.

Input

First line consists of a number N - the number of test cases (0 < N < 10). Each next line (which represents a test case) contains an English text (0 <= length(text) < 999).

Output

For each test case, print the English letters (case insensitive) sorted in descending order by the number of times they appeared in the text. In case two or more letters have the same frequency, sort them in ascending order (a to z).

Sample

Input

```
2
This                is                a                 text.
Looks like it is.
```

Output

```
tisaehx
ikloset
```