

Exercises: Algorithms Exam Preparation

This document defines the **in-class exercises** assignments for the ["Algorithms" course @ Software University](#). Submit your solutions in [this contest](#).

Problem 1. Parenthesis

Given n pairs of parentheses, write a program to generate all combinations of well-formed parentheses.

The order of output does not matter.

Input	Output
3	((())) (()()) (())() ()(()) ()()()

Problem 2. Guitar

Bobi is a guitar player and he is going to play a concert. He doesn't like to play all the songs at the same volume, so he decides to **change the volume level** of his guitar before each new song. Before the concert begins, he makes a **list of the number of intervals** he will be changing his volume level by before each song. For each volume change, he will decide whether to **add that number of intervals to the volume or subtract it**.

You are given a list of integers **C**, the i -th element of which is the number of intervals Bobi will change his volume by before the i -th song. You are also given an integer **B**, the initial volume of Bobi's guitar, and an integer **M**, the highest possible volume setting of Bobi's guitar. Bobi cannot change the volume of his guitar to a level above **M** or below 0 (but exactly 0 and exactly **M** is possible). Your program should print the maximum volume Bobi can use to play the last song. If there is no way to go through the list without exceeding **M** or going below 0, print -1.

Input

The input data should be read from the console.

The elements of the list **C** will be on the first input line separated by a comma and an interval (", ").

On the second line there will be the number **B** and on the third line there will be the number **M**.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The output data should be printed on the console.

On the only output line you should print -1 or the maximum volume **Bobi** can use to play the last song.

Constraints

- **C** will contain between 1 and 50 elements, inclusive.
- In 95% of the tests cases **C** will contain no less than 34 elements.
- Each element of **C** will be between 1 and **M**, inclusive.
- **M** will be between 1 and 1000, inclusive.

- **B** will be between 0 and **M**, inclusive.
- Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

Examples

Input	Output
5, 3, 7 5 10	10

Input	Output
15, 2, 9, 10 8 20	-1

Input	Output
74, 39, 127, 95, 63, 140, 99, 96, 154, 18, 137, 162, 14, 88 40 243	238

Problem 3. Towns

You are given **N** towns and you should find a path in them that satisfies few conditions. In each of the towns there are fixed number of **citizens** (between 1 and 1,000,000,000, inclusive).

The towns can be visited **only in the order they are given** on the console. You are not obligated to start the traversing from the first town. You can also **skip some of the towns** but you are not allowed to visit a town more than once.

For example if we have towns Sofia, Plovdiv, Varna and Burgas **you can visit**:

- Sofia, Varna, Burgas (by skipping Plovdiv)
- Plovdiv, Burgas (by starting from Plovdiv and skipping Varna)
- Sofia, Plovdiv, Varna, Burgas (by visiting every town)
- etc.

You are **not allowed** to visit the towns in wrong order. For example:

- Varna, Sofia is **not** allowed because Varna should be visited after Sofia in the given order
- Sofia, Varna, Sofia is **not** allowed because you are visiting Sofia for a second time
- etc.

You should satisfy **one more condition** when traversing towns. If you divide the path into 2 sub-paths that share one common town (part of the original path) such that the last town of the first path is the first town of the second path, then all number of citizens in the first path should be in ascending order and all number of citizens (increasing number of citizens) in the second path should be in descending order (decreasing number of citizens). Yeah.

For example look at the first example below. The path Pleven, Burgas, Varna, Sofia, Ruse, StaraZagora satisfies this condition (in fact, it satisfies all conditions) because we can divide the path into 2 sub-paths (first: Pleven, Burgas, Varna, Sofia; second: Sofia, Ruse, StaraZagora) and the first path has increasing number of citizens and the second path has decreasing number of citizens.

Write a program that finds **the longest path that satisfies the given conditions**.

Input

The input data should be read from the console.

The number of towns **N** will be given on the first input line.

On each of the next **N** lines there will be information for one of the towns in the order they can be traversed. If you split the line by a single space (' ') the first part of the line will contain the number of citizens in the town, and the second part of the line will contain the name of the town. See the examples below.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The output data should be printed on the console.

On the only output line write the longest path that satisfies the given conditions.

Constraints

- **N** will be an integer between 1 and 1000, inclusive.
- The name of each town will contains only Latin letters (both lowercase ('a'-'z') and uppercase ('A'-'Z') are allowed)
- The length of each town name will be between 1 and 20 characters.
- Each town name will be unique.
- Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

Examples

Input	Output	Explanation
8 108214 Pleven 339077 Plovdiv 200612 Burgas 334688 Varna 1241396 Sofia 92162 Sliven 151951 Ruse 137907 StaraZagora	6	The longest path that satisfies all conditions contains these 6 cities: Pleven (first town) Burgas (bigger than Pleven) Varna (bigger than Burgas) Sofia (bigger than Varna) Ruse (smaller than Sofia) StaraZagora (smaller than Ruse) (The path can be cut in 2 paths satisfying citizens count condition. Sofia is the town that divides the path into 2 other paths)
4 19906 NikiTown 19832 EvlogiTown 19894 IvoTown 19896 DonchoTown	3	You can start from EvlogiTown, then go in IvoTown and then DonchoTown. If we divide the path into 2 sub-paths <ul style="list-style-type: none"> • EvlogiTown, IvoTown and DonchoTown • DonchoTown They satisfy the ascending-descending condition.
3 2 HaxorTown 2 LeetTown 2 RoxxorTown	1	You can visit any of the 3 towns, but cannot move any further because of the citizens count traversing requirements

Problem 4. Fast and Furious

The ministry of interior recently deployed a system of **traffic cameras** on different locations on the roads. Some **pairs of cameras** are connected with direct road and you are given the **distance** and the **speed limit** between them.

You are given the **records from the traffic cameras** on the road. Each camera takes photos of car license plate numbers and **reports the numbers and the time of observation**. Your task is to find which cars are **speeding**.

A car travelling between two arbitrary cameras **A** and **B** on the road is **speeding** if it takes the distance between these cameras for **less time than the minimum possible** within the allowed speed limits. Note that many routes may exist between **A** and **B** and each of them can be passed for different times depending on the distances and speed limits for the roads in each route. We assume that the drivers always take the fastest route between two cameras.

Input

- The input is read from the console.
- At the first line the word **"Roads:"** stays.
- The next few lines hold **pairs of camera names**, the **distance** between them and the **speed limit** (in km/h) between them. The camera names and maximum speed are separated by a single space. Example:

CameraSofia CameraPleven 133.35 140

- At the next line the word **"Records:"** stays.
- The next few lines hold a sequence of **camera records**. Each record consist of a **camera name**, a **license plate number** and a **time** in 24-hour format (**hh:mm:ss**), separated by a space. Example:

CameraSofia CA1111AA 12:56:12

- The last line holds the word **"End"** only.

Output

- Print the **license plate numbers of all speeding cars** in alphabetical order, each on separate line. Example:

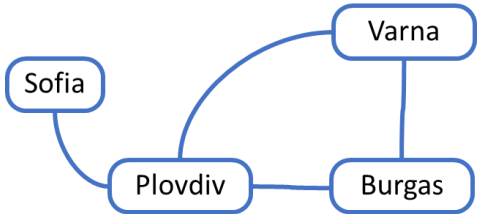
CA1111AA
CA1212BB
CHY0L0428

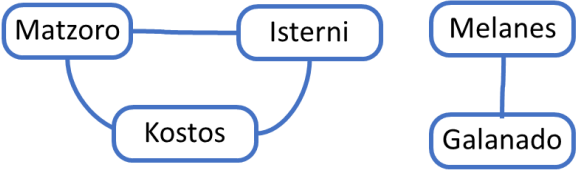
Constraints

- All **camera names** consist of letters and digits.
- All **license plate numbers** consist of letters and digits.
- The **distances** and **speed limits** are real numbers in the range [1...10 000].
 - The symbol **"."** is used as decimal separator.
- The **number of roads** is in the interval [1; 1 000].
- The number of **camera records** is in the interval [1; 10 000].
- All data is collected on the **same day**.
- Cameras collect their records in unspecified order.
- Time limit: **200 ms**. Allowed memory: **24 MB**.

Sample Input and Output

Input	Visualization and Comments
Roads :	The cameras are connected by roads like shown below:

<p>Sofia Plovdiv 145.4 90 Plovdiv Varna 361.4 120.5 Varna Burgas 114.95 30 Burgas Plovdiv 252.9 42</p> <p>Records: Varna CA1234AA 19:48:25 Burgas B4732AH 19:38:50 Sofia CA1234AA 08:32:18 Plovdiv A777777 15:28:56 Varna SP33D 02:24:18 Burgas A777777 18:42:15 Plovdiv CA1234AA 15:32:18 Sofia SP33D 04:32:51 Varna B4732AH 08:18:36</p> <p>End</p>	 <p>Car "A777777" is speeding between Plovdiv and Burgas.</p> <ul style="list-style-type: none"> It takes the distance of 252.9 km from Plovdiv to Burgas for 3:13:19 hours (18:42:15 @ Burgas - 15:28:56 @ Plovdiv) \approx 3.222 h. The minimum time within the allowed maximum speed limits from Plovdiv to Burgas is 252.9 km / 42 km/h \approx 6.02 hours. The car was speeding because the driving time (3.222 hours) < the minimum possible time within the speed limits (6.02 hours). <p>Car "SP33D" is speeding between Varna and Sofia.</p> <ul style="list-style-type: none"> It takes the distance between Varna and Sofia for 2:08:33 hours (04:32:51 @ Sofia - 02:24:18 @ Varna) \approx 2.1425 hours. Two routes exist from Varna to Sofia: <ul style="list-style-type: none"> For the route Varna \rightarrow Plovdiv \rightarrow Sofia the minimum time within the speed limit is 2.999 hours (Varna \rightarrow Plovdiv) + 1.616 (Plovdiv \rightarrow Sofia) \approx 4.615 hours. For the route Varna \rightarrow Burgas \rightarrow Plovdiv \rightarrow Sofia the minimum time within the speed limit is \approx 11.469 hours. The car was speeding because the driving time (2.1425 hours) < the minimum possible time within the speed limits (4.615 hours).
Output	
A777777 SP33D	

Input	Visualization and Comments
<p>Roads: Matzoro Isterni 128.55 50 Matzoro Kostos 87.25 48.5 Isterni Kostos 100 40.52 Melanes Galanado 230.5 50</p> <p>Records: Isterni AOM5973 13:20:11 Matzoro IBK5674 08:35:12 Matzoro AHI1278 08:35:12 Galanado IBK5674 18:20:35 Kostos COM1515 05:38:02 Galanado COM1515 08:40:15 Isterni IBK5674 14:28:30 Melanes COM1515 22:31:50 Kostos AOM5973 12:46:21 Kostos COM1515 18:56:10</p> <p>End</p>	<p>The cameras are connected by roads like shown below:</p>  <p>Car "AOM5973" is speeding between Kostos and Isterni:</p> <ul style="list-style-type: none"> It takes the distance of 100 km from Kostos to Isterni for 0:33:50 hours (13:20:11 @ Isterni - 12:46:21 @ Kostos) \approx 0.564 h. The minimum time within the allowed maximum speed limits from Kostos to Isterni is 100 km / 40.52 km/h \approx 2.468 hours. The car was speeding because the driving time (0.564 hours) < the minimum possible time within the speed limits (2.468 hours).
Output	
AOM5973	