Problem Statement 1 :

In this problem you have to implement the linked list. Linked list should support three types of operations :

1. Insert element in linked list at position 'p'

   Operation code : 1

   Input Format:

   Each query for insertion operation contains three space separated integers , ‘o’, ‘p’ and ‘e’ where :

   ‘o’ is operation code

   ‘p’ is the position at which the operation is to be performed

   ‘e’ is the element to be inserted.

2. Delete element in linked list at position 'p'

   Operation code : 2

   Each query for deletion operation contains three space separated integers , ‘o’ and ‘p’ where :

   ‘o’ is operation code

   ‘p’ is the position at which the operation is to be performed

3. Update element in linked list at position 'p'

   Operation code: 3

   Each query for updation operation contains three space separated integers , ‘o’, ‘p’ and ‘e’ where :

   ‘o’ is operation code

   ‘p’ is the position at which the operation is to be performed

   ‘e’ is the element to be replaced with the original element.

Input:

First line contains t which specifies the number of test cases. Then it is followed by next ‘t’ test cases. First line of each test case contains an integer ‘q’, which specifies the number of operations to be performed in the linked list. Next q lines in each test case contain queries for the operation to be performed.

Output:

After performing each operation print the updated linked list in a new line as illustrated in sample output.

Note:

There will be no invalid operation or query given as input.

Sample Input:

1

6

1 0 3

1 0 35

1 1 4

3 1 5

3 2 9

2 1

Sample Output:

3

35->3

35->4->3

35->5->3

35->5->9

35->9

Problem Statement 2:

Let us see how search engines work. Consider the following simple auto complete feature. When you type some characters in the text bar, the engine automatically gives best matching options among it's database. Your job is simple. Given an incomplete search text, output the best search result.

Each entry in engine's database has a priority factor attached to it. We consider a result / search suggestion best if it has maximum weight and completes the given incomplete search query. For each query in the input, print the maximum weight of the string in the database, that completes the given incomplete search string. In case no such string exists, print **-1**.

Input

First line contains two integers **n** and **q**, which represent number of database entries and number of search queries need to be completed. Next **n** lines contain a string **s** and an integer **weight**, which are the database entry and it's corresponding priority.

Next **q** lines follow, each line having a string **t**, which needs to be completed.

Output

Output **q** lines, each line containing the maximum possible weight of the match for given query, else **-1**, in case no valid result is obtained.

Constraints

$1 \leq$ **n**, **weight**, **len(s)**, **len(t)** $\leq 10^3$

$1 \leq$ **q** $\leq 10^3$

total length of all strings in database entries $\leq 10^4$

total length of all query strings $\leq 10^4$

Sample Input:

3 1

hellodevansh 10

hellopramod 9

helldsdhf 11

hello

Sample Output:

10

Problem Statement 3:

You are given to make a database of student details. This database would contain three tables. These tables would be given as input. The first table is Course table, second one is Student table and the third one is Grades table. You have to store these tables in array of structures. The table structure is provided below:

Course Table:
Course Id: Data Type: Non-Negative integer
Course Name: Data Type: String

Student Table:
Student Id: Data Type: Non-Negative integer
Student Name: Data Type: String

Marks Table:
Student Id: Data Type: Non-Negative integer
Course Id: Data Type: Non-Negative integer
Marks: Data Type: Non-Negative Integer

For this table you have to perform three types of queries. Each query has its operation code. Operation code would be given as input. You have to perform the query for the corresponding operation code.

Query Types:

1. To find the number of students who have taken a particular course
   Operation code: 1
   Input: As input course name would be only provided.
   Output: Output the number of students for this query in a new line

2. To find the number of students who have taken course A and course B
   Operation code: 2
   Input: Two space separated strings would be given. First string represents course A and the second one represents course B.
   Output: Output the number of students for this query in a new line
3. To find the student name with highest marks in course A
   Operation code: 3
   Input: As input a single string A is given (course name).
   Output: Print the student name with highest marks in course A

Input:

First line contains t which specifies the number of test cases. Structure of input for next t test cases is as follows:

First line of each test case contains four space separated integers u, v, x and n. u is the number of entries in the Course table. v is the number of entries in the Student table. x is the number of entries in the Marks table. n specifies the number of queries to be performed.

After this u entries for the course table is given. Each entry of course table contains two space separated inputs “Course Id” and “Course Name” in the new line.
After this v entries for the course table is given. Each entry of course table contains
two space separated inputs “Student Id” and “Student Name” in the new line.
After this x entries for the course table is given. Each entry of course table contains
three space separated inputs “Course Id”, “Student Id” and “Marks” in the new line.
After this n queries are given in the new line. For each type of query, the query structure is given already.

Output:

For each test case, output the answer of n queries in newline as specified in the query structure.

Constraints:

1<=t<=10
1<=u, v, x, n<=20
Student Id, Course Id, Course Name are unique.
It is not necessary that Student Id and Course Id are consequtive.
In a particular course only student has highest marks.

Sample Input:

```
1
2 3 4 5
1 Maths
3 CP
5 DLD
1 Prasanth
2 Abhishek
1 1 55
1 3 67
2 3 89
2 1 98
1 Maths
2 Maths CP
3 Maths
3 CP
2 Maths DLD
```

Sample Output:

```
2
2
Abhishek
Abhishek
0
```

Problem Statement 4

There is a planet with 2 people. On their planet, it is believed that their friendship will last forever, if they merge their favorite strings and etch it on the surface of a stone. So we will mingle their favorite strings. The lengths of their favorite strings is same (say n). Mingling two strings, P=p 1 p 2 ...p n and Q=q 1 q 2 ...q n , both of length n, will result in creation of a new string R of length 2×n. It will have the following structure:

$R=p_1q_1p_2q_2...p_nq_n$

Input

First line specifies the number of test cases t. Second line specifies n, the length of both strings.

Output
For each test case output the mingled string of length n×m in a new line.

Constraints
1<=t<=10
1<=n<=20
You have to mingle strings using recursion.

Sample Input:
1
2 3
Sad
sdf

Sample Output:
Ssaddf

Problem Statement 5:

Given a array of strings you have to sort them according to their length. The length of strings will be unique.

Input:
First line specifies 't', the number of test cases. In each of the t test cases, first line contains a single integer 'n' which specifies the size of array of strings. Next 'n' lines contain 'n' strings as illustrated in the sample output.

Output:
For each test case print the sorted array of strings in lexicographic order as illustrated in sample output.

Constraints:
1<=t<=10
1<=n<=10

Sample Input:

```
2
4
ajhdjs
sjdh
djhss
adjshss
4
asssdss
sdsfs
sdsdss
ssss
```

Sample Output:

```
adjshss
ajhdjs
djhss
sjdh
asssdss
sdsdss
sdsfs
ssss
```

Problem Statement 6 (Bonus):

Given a string print all it's permutations in lexicographic order.

Input:

First line specifies the string s to be permuted.

Output:

For the given string 's', output all the possible permutations of the string 's'.

Constraints:

1<=len(s)<=7

Sample Input:

dgf

Sample Output:

dfg

dgf

fdg

fgd

gdf

gfd