Codeforces Round 1043 (Div. 3)

A. Homework

1 second, 256 megabytes

Vlad and Dima have been assigned a task in school for their English class. They were given two strings a and b and asked to append all characters from b to string a in any order. The guys decided to divide the work between themselves and, after lengthy negotiations, determined who would add each character from string b to a.

Due to his peculiarities, Vlad can only add characters to the beginning of the word, while Dima can only add them to the end. They add characters in the order they appear in string b. Your task is to determine what string Vlad and Dima will end up with.

Input

Each test consists of several test cases. The first line contains a single integer t $(1 \le t \le 1000)$ — the number of test cases. The description of the test cases follows.

The first line contains an integer n $(1 \le n \le 10)$ — the length of the string a.

The second line contains the string a, consisting of lowercase letters of the English alphabet.

The third line contains an integer m $(1 \le m \le 10)$ — the length of the strings b and c.

The fourth line contains the string b, consisting of lowercase letters of the English alphabet.

The fifth line contains the string c, consisting of the characters 'V' and 'D' — the distribution of the characters of string b between Dima and Vlad. If c_i = 'V', then the i-th letter is added by Vlad; otherwise, it is added by Dima.

Output

For each test case, output the string that will result from Dima and Vlad's work.

input

```
4
2
ot
2
ad
DV
3
efo
7
rdcoecs
DVDVDVD
3
aca
4
bbaa
DVDV
3
biz
4
abon
VVDD
```

output

```
dota
codeforces
abacaba
babizon
```

In the first test case, there is initially a string ot. Then Dima appends the character a to the end of the string, resulting in ota, and Vlad appends the last character, resulting in $dota$.

In the second test case, the string will change as follows: $efo \to efor \to defor \to deforc \to odeforc \to odeforce \to codeforce \to codeforces$

In the third test case: $aca \to acab \to bacab \to bacaba \to abacaba$

In the fourth test case: $biz \to abiz \to babiz \to babizo \to babizon$

B. The Secret Number

2 seconds, 256 megabytes

Vadim has thought of a number x. To ensure that no one can guess it, he appended a positive number of zeros to the right of it, thus obtaining a new number y. However, as a precaution, Vadim decided to spread the number $n = x + y$. Find all suitable x that Vadim could have thought of for the given n.

Input

Each test consists of several test cases. The first line contains a single integer t $(1 \le t \le 10^4)$ — the number of test cases. The following lines describe the test cases.

In a single line of each test case, there is an integer n — the number spread by Vadim $(11 \le n \le 10^{18})$.

Output

For each number n, output 0 if there are no suitable x. Otherwise, output the number of suitable x, followed by all suitable x **in ascending order**.

input

```
5
1111
12
55
999999999999999999
1000000000000000000
```

output

```
2
11 101
0
1
5
3
999999999 999000999000999 90909090909090909
0
```

In the first sample, to 11 one can append two zeros to the right, then $11 + 1100 = 1111$, and to 101 one can append one zero to the right, then $101 + 1010 = 1111$.

In the second sample, it is impossible to obtain 12 through the described actions.

C1. The Cunning Seller (easy version)

2 seconds, 256 megabytes

This is the easy version of the problem. The easy version differs from the hard one in that it requires determining the minimum cost with the least number of deals, while the hard version requires determining the minimum cost with a limited number of deals.

After the cunning seller sold three watermelons instead of one, he decided to increase his profit — namely, he bought even more watermelons. Now he can sell 3^x watermelons for $3^{x+1} + x \cdot 3^{x-1}$ coins, where x is a non-negative integer. Such a sale is called a deal.

A calculating buyer came to him, but he has critically little time. Because of this, he wants to buy exactly n watermelons, making the **least** possible number of deals.

The buyer is in a hurry and has therefore turned to you to determine the minimum number of coins he must pay the seller for n watermelons, considering that he will make the **least** possible number of deals.

Input

The first line contains an integer t $(1 \le t \le 10^4)$ — the number of test cases. The description of each test case follows.

In a single line of each test case, there is one integer n $(1 \le n \le 10^9)$ — how many watermelons need to be bought.

Output

For each test case, output a single integer — the minimum cost of the watermelons.

input

```
7
1
3
8
2
10
20
260010000
```

output

```
3
10
26
6
36
72
2250964728
```

Note that there is no point in buying more watermelons than needed, so we won't consider deals where there are more watermelons than necessary.

Let's consider the costs of the first two deal options:

Deal A: 1 watermelon — 3 coins.

Deal B: 3 watermelons — 10 coins.

In the first sample, the only way to buy 1 watermelon is to use Deal A, so the answer is 3.

In the second sample, you can buy 3 watermelons with a single Deal B for 10 coins.

In the third sample, you can make 2 Deals A and 2 Deals B, which will cost a total of 26 coins. If we make 3 deals, we can get 3, 5, 7, or 9 watermelons. If we make fewer than 3 deals, we will get no more than 6 watermelons, which means it is impossible to buy 8 watermelons for less than 4 deals.

C2. The Cunning Seller (hard version)

2 seconds, 256 megabytes

This is the hard version of the problem. The easy version differs from the hard one in that it requires determining the minimum cost with the least number of deals, while the hard version requires determining the minimum cost with a limited number of deals.

After the cunning seller sold three watermelons instead of one, he decided to increase his profit — namely, he bought even more watermelons. Now he can sell 3^x watermelons for $3^{x+1} + x \cdot 3^{x-1}$ coins, where x is a non-negative integer. Such a sale is called a deal.

A calculating buyer came to him, but he has little time, so the buyer can make no more than k deals and plans to buy exactly n watermelons.

The buyer is in a hurry and has therefore turned to you to determine the minimum number of coins he must pay the seller for n watermelons if he makes no more than k deals. If it is impossible to buy exactly n watermelons while making no more than k deals, output -1.

Input

The first line contains an integer t $(1 \le t \le 10^4)$ — the number of test cases. The description of each test case follows.

In a single line of each test case, there are two integers n and k $(1 \le n, k \le 10^9)$ — how many watermelons need to be bought and how many deals can be made.

Output

For each test case, output a single integer — the minimum cost of the watermelons or -1 if it is impossible to buy the watermelons while meeting all the conditions.

input

```
8
1 1
3 3
8 3
2 4
10 10
20 14
3 2
9 1
```

output

```
3
9
-1
6
30
63
10
33
```

Note that there is no point in buying more watermelons than needed, so we will not consider deals where there are more watermelons than necessary.

Let's consider the costs of the first two deal options:

Deal A: 1 watermelon — 3 coins.

Deal B: 3 watermelons — 10 coins.

In the first sample, the only way to buy 1 watermelon is to use Deal A, so the answer is 3.

In the second sample, you can buy 3 watermelons either with Deal B for 10 coins or with three Deal A for 9 coins, so the answer is 9.

In the third sample, there are the following options for 3 deals:

3 Deals A — 3 watermelons.

2 Deals A and 1 Deal B — 5 watermelons.

1 Deal A and 2 Deals B — 7 watermelons.

3 Deals B — 9 watermelons.

It can be seen that it is impossible to buy **exactly** 8 watermelons.

D. From 1 to Infinity

1.5 seconds, 256 megabytes

Vadim wanted to understand the infinite sequence of digits that consists of the positive integers written consecutively from 1 to infinity. That is, this sequence looks like $123456789101112131415\ldots$

To avoid looking into infinity, Vadim cut this sequence at the k-th digit and discarded everything after it. Thus, exactly k digits remained in the sequence. Help him find the sum of the digits in the remaining sequence.

Input

Each test case consists of several test cases. The first line contains a single integer t $(1 \le t \le 2 \cdot 10^4)$ — the number of test cases. The following lines describe the test cases.

In a single line of each test case, there is an integer k — the number of digits in the remaining sequence $(1 \le k \le 10^{15})$.

Output

For each given k, output the sum of the digits in the sequence of length k.

input

```
6
5
10
13
29
1000000000
1000000000000000
```

output

```
15
46
48
100
4366712386
4441049382716054
```

In the first sample, the remaining sequence will be 12345.

In the second sample, the remaining sequence will be 1234567891.

In the third sample, the remaining sequence will be 1234567891011.

E. Arithmetics Competition

3 seconds, 256 megabytes

In the arithmetic competition, participants need to achieve the highest possible sum from the cards they have. In the team *"fst_ezik"*, Vadim has n cards with numbers a_i, and Kostya has m cards with numbers b_i. In each of the q rounds, they want to win, but this time the rules of the competition are slightly different from the usual ones.

In each round, the participants are given three numbers x_i, y_i, and z_i. The team *"fst_ezik"* must choose exactly z_i cards from all the cards they have, but Vadim can choose no more than x_i cards from his set, and Kostya can choose no more than y_i cards from his set. Help them find the highest possible sum for each of the q rounds.

Input

Each test consists of several test cases. The first line contains a single integer t $(1 \le t \le 10^4)$ — the number of test cases. The descriptions of the test cases follow.

In the first line of each test case, three integers n, m, q are given $(1 \le n, m \le 2 \cdot 10^5, 1 \le q \le 10^5)$ — the number of cards Vadim has, the number of cards Kostya has, and the number of rounds in the competition.

The second line contains n integers a_i — the numbers on Vadim's cards $(1 \le a_i \le 10^9)$.

The third line contains m integers b_i — the numbers on Kostya's cards $(1 \le b_i \le 10^9)$.

The following q lines describe the rounds with three integers x_i, y_i, z_i $(0 \le x_i \le n, 0 \le y_i \le m, 0 \le z_i \le x_i + y_i)$ — the limit on the number of cards Vadim can choose, the limit on the number of cards Kostya can choose, and the number of cards they need to select together.

It is guaranteed that the sum of n across all test cases does not exceed $2 \cdot 10^5$, the sum of m across all test cases does not exceed $2 \cdot 10^5$, and the sum of q across all test cases does not exceed 10^5.

Output

For each test case, output q numbers — the highest possible sum for the corresponding round.

input

```
4
3 4 5
10 20 30
1 2 3 4
0 0 0
3 4 7
3 4 4
1 4 4
2 2 4
5 5 2
500000000 300000000 100000000 900000000 700000000
800000000 400000000 1000000000 600000000 200000000
1 4 3
5 2 6
4 4 1
100 100 20 20
100 100 20 20
4 4 5
3 3 6
2 363 711
286 121 102
1 1 1
3 1 1
1 2 0
1 3 2
0 1 0
3 3 3
```

output

```
0
70
64
39
57
2700000000
4200000000
420
711
711
0
997
0
1360
```

F. Rada and the Chamomile Valley

3 seconds, 512 megabytes

Yesterday, Rada found a portal that can transport her to the Chamomile Valley and back. Rada's happiness knew no bounds, but it didn't last long — she suddenly realized that she didn't know where and when any of the Smeshariki would be.

The Chamomile Valley consists of n houses and m lanes connecting the houses. The lanes are numbered from 1 to m. You can walk along the lanes in both directions. It is known that from any house, you can reach any other house via the lanes, and there is no lane connecting a house to itself. Moreover, any two houses are connected by at most one lane.

Rada knows that the Smeshariki walk every day from house number 1 to house number n, but she doesn't know which specific lanes they will take. Rada will be in the Chamomile Valley on each of the next q days. On the k-th day, she will be at house number c_k.

Since Rada does not know which specific lanes the Smeshariki will take, she is only interested in those lanes that they will **definitely** use. To ensure she does not miss any of them, she wants to know the index of the nearest such lane on each day. Rada is too busy strolling through the Chamomile Valley, so she asks you to help her determine the required lane indices.

The distance from house c to the lane connecting houses a and b is defined as the minimum of $\rho(a, c)$ and $\rho(b, c)$, where $\rho(a, b)$ is the minimum number of lanes needed to reach house number b starting from house number a.

Input

The first line of the input contains an integer t $(1 \le t \le 10^4)$ — the number of test cases. The description of each test case follows.

The first line contains two integers n and m $(1 \le n \le 2 \cdot 10^5$, $n - 1 \le m \le \min(\frac{n \cdot (n-1)}{2}, 2 \cdot 10^5))$ — the number of houses and lanes, respectively.

The next m lines contain two integers $u \ne v$ $(1 \le u, v \le n)$ — a lane connecting houses numbered u and v. The lanes are given in order of numbering, that is, the description of the first lane comes first, followed by the second, third, and so on up to the m-th lane.

Next, an integer q $(1 \le q \le 2 \cdot 10^5)$ is given — the number of days Rada will be walking in the Chamomile Valley.

The next q lines each contain a single integer c $(1 \le c \le n)$ — the house at which Rada will be on that day.

It is guaranteed that from any house, you can reach any other house by only using the lanes, and there are no lanes from a house to itself, and any two houses are connected by at most one lane.

It is guaranteed that the sum of n across all test cases does not exceed $2 \cdot 10^5$, the sum of m across all test cases does not exceed $2 \cdot 10^5$, and the sum of q across all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the answer for each of the days. If there are multiple suitable lanes on any of the days, output the lane with the **smallest** index among the suitable ones. If there are no required lanes, output -1.

input

```
3
3 3
1 2
2 3
3 1
1
1
5 4
1 2
2 3
3 4
4 5
3
1
2
3
7 6
1 2
1 5
2 3
3 4
5 7
6 7
7
1
2
3
4
5
6
7
```

output

```
-1
1 1 2
2 2 2 2 2 5 5
```

input

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

output

```
3 3 3 3 3
1 6
```

In all further explanations, we denote the transition from house a to house b via the lane numbered c as $a \xrightarrow{c} b$.

In the first sample, from house 1 to house 3, you can reach it via at least the following paths:

$$1 \xrightarrow{3} 3$$

$$1 \xrightarrow{1} 2 \xrightarrow{2} 3$$

As we can see, these two paths do not share any common lanes, which means there are no suitable lanes.

In the second sample, it can be noted that there is a unique path from 1 to n:

$$1 \xrightarrow{1} 2 \xrightarrow{2} 3 \xrightarrow{3} 4 \xrightarrow{4} 5$$

As can be seen, the answer for the v-th vertex is $\max(v-1,\ 1)$.

G. Famous Choreographer

3 seconds, 512 megabytes

As all programmers know, there are $n \times m$ ballerinas performing in a ballet, and their arrangement can be represented as a table with n rows and m columns. Each ballerina performs one of 26 movements, which can be described by one of the English letters.

Choreographer Vadim wants to dispel this myth. To do this, he wants to stage a show in which all the ballerinas gracefully move to the opposite side of the stage from their starting positions. Programmers will find it easier to understand this movement as a 180° rotation of the table. To maintain the sequence of visual storytelling in the ballet, the ballerinas perform this movement instantaneously, without stopping their movements, and the final arrangement is identical to the initial one.

Unfortunately, Vadim understands that with the current performance and the already planned arrangement of the ballerinas, such a maneuver will not be possible. Therefore, he is ready to invite more ballerinas to the performance. They can perform any movement and occupy any position, but they cannot stand between those already participating in a ballet. The most important thing is that in the end, a rectangular table is formed, possibly larger than the original one. Additionally, it is essential that **at least one** ballerina from the original arrangement moves to the position of one of the other ballerinas from the original arrangement or remains in her place. Please advise Vadim on the smallest number of ballerinas he will need to invite.

Input

Each test consists of several test cases. The first line contains a single integer t $(1 \le t \le 10^5)$ — the number of test cases. The following lines describe the test cases.

In the first line of each set, two integers n and m are given — the number of rows and the number of columns of the table $(1 \le n, m \le 10^6, 1 \le n \cdot m \le 10^6)$.

The next n lines of length m describe the movements of the ballerinas — the ballerina in the i-th row and j-th column performs the movement f_{ij}, where f_{ij} — is a lowercase English letter.

It is guaranteed that the sum of $n \cdot m$ across all test cases does not exceed 10^6.

Output

For each test case, output the minimum number of ballerinas that Vadim will need to invite.

input

```
6
2 3
hey
hey
3 3
abc
def
ghi
3 2
af
fa
te
1 1
x
3 3
uoe
vbe
mbu
2 3
hyh
kop
```

output

```
4
16
2
0
11
3
```

In the first sample, you can invite 4 ballerinas and arrange them as follows:

h e y **e** **h**

h e y **e h**

In the second sample, you can invite 16 ballerinas and arrange them like this:

j k a b c

k j d e f

i h g h i

f e d j k

c b a k j

In the third sample, you can invite 2 ballerinas and arrange them like this:

e t

a f

f a

t e

Here, the invited ballerinas are marked in bold.