A. Average Height

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Sayaka Saeki is a member of the student council, which has n other members (excluding Sayaka). The i-th member has a height of a_i millimeters.

It's the end of the school year and Sayaka wants to take a picture of all other members of the student council. Being the hard-working and perfectionist girl as she is, she wants to arrange all the members in a line such that the amount of *photogenic* consecutive pairs of members is **as large as possible**.

A pair of two consecutive members u and v on a line is considered *photogenic* if their average height is an integer, i.e. $\frac{a_u + a_v}{2}$ is an integer.

Help Sayaka arrange the other members to **maximize** the number of photogenic consecutive pairs.

Input

The first line contains a single integer t ($1 \le t \le 500$) — the number of test cases.

The first line of each test case contains a single integer n ($2 \le n \le 2000$) — the number of other council members.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 2 \cdot 10^5$) — the heights of each of the other members in millimeters.

It is guaranteed that the sum of n over all test cases does not exceed 2000.

Output

For each test case, output on one line n integers representing the heights of the other members in the order, which gives the largest number of photogenic consecutive pairs. If there are multiple such orders, output any of them.

Example

input

```
4
3
1 1 2
3
1 1 1
8
10 9 13 15 3 16 9 13
2
18 9
```

output

```
1 1 2
1 1 1
13 9 13 15 3 9 16 10
9 18
```

Note

In the first test case, there is one photogenic pair: $(1, 1)$ is photogenic, as $\frac{1+1}{2} = 1$ is integer, while $(1, 2)$ isn't, as $\frac{1+2}{2} = 1.5$ isn't integer.

In the second test case, both pairs are photogenic.

B. TMT Document

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The student council has a shared document file. Every day, some members of the student council write the sequence TMT (short for Towa Maji Tenshi) in it.

However, one day, the members somehow entered the sequence into the document at the same time, creating a jumbled mess. Therefore, it is Suguru Doujima's task to figure out whether the document has malfunctioned. Specifically, he is given a string of length n whose characters are all either T or M, and he wants to figure out if it is possible to partition it into some number of disjoint subsequences, all of which are equal to TMT. That is, each character of the string should belong to exactly one of the subsequences.

A string a is a subsequence of a string b if a can be obtained from b by deletion of several (possibly, zero) characters.

Input

The first line contains an integer t $(1 \le t \le 5000)$ — the number of test cases.

The first line of each test case contains an integer n $(3 \le n < 10^5)$, the number of characters in the string entered in the document. It is guaranteed that n is divisible by 3.

The second line of each test case contains a string of length n consisting of only the characters T and M.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, print a single line containing YES if the described partition exists, and a single line containing NO otherwise.

Example

input

```
5
3
TMT
3
MTT
6
TMTMTT
6
TMTTTT
6
TTMMTT
```

output

```
YES
NO
YES
NO
YES
```

Note

In the first test case, the string itself is already a sequence equal to TMT.

In the third test case, we may partition the string into the subsequences **TM**TM**T**T. Both the bolded and the non-bolded subsequences are equal to TMT.

C. The Sports Festival

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

The student council is preparing for the relay race at the sports festival.

The council consists of n members. They will run one after the other in the race, the speed of member i is s_i. The *discrepancy* d_i of the i-th stage is the difference between the maximum and the minimum running speed among the first i members who ran. Formally, if a_i denotes the speed of the i-th member who participated in the race, then $d_i = \max(a_1, a_2, \dots, a_i) - \min(a_1, a_2, \dots, a_i)$.

You want to minimize the sum of the discrepancies $d_1 + d_2 + \dots + d_n$. To do this, you are allowed to change the order in which the members run. What is the minimum possible sum that can be achieved?

Input

The first line contains a single integer n $(1 \le n \le 2000)$ — the number of members of the student council.

The second line contains n integers $s_1, s_2, \dots, s_n$ $(1 \le s_i \le 10^9)$ – the running speeds of the members.

Output

Print a single integer — the minimum possible value of $d_1 + d_2 + \dots + d_n$ after choosing the order of the members.

Examples

input

```
3
3 1 2
```

output

```
3
```

input

```
1
5
```

output

```
0
```

input

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

output

```
11
```

input

```
6
104 943872923 6589 889921234 1000000000 69
```

output

```
2833800505
```

Note

In the first test case, we may choose to make the third member run first, followed by the first member, and finally the second. Thus $a_1 = 2$, $a_2 = 3$, and $a_3 = 1$. We have:

- $d_1 = \max(2) - \min(2) = 2 - 2 = 0$.
- $d_2 = \max(2, 3) - \min(2, 3) = 3 - 2 = 1$.
- $d_3 = \max(2, 3, 1) - \min(2, 3, 1) = 3 - 1 = 2$.

The resulting sum is $d_1 + d_2 + d_3 = 0 + 1 + 2 = 3$. It can be shown that it is impossible to achieve a smaller value.

In the second test case, the only possible rearrangement gives $d_1 = 0$, so the minimum possible result is 0.

D. Binary Literature

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A bitstring is a string that contains only the characters `0` and `1`.

Koyomi Kanou is working hard towards her dream of becoming a writer. To practice, she decided to participate in the *Binary Novel Writing Contest*. The writing prompt for the contest consists of three bitstrings of length $2n$. A valid novel for the contest is a bitstring of length at most $3n$ that contains **at least two** of the three given strings as subsequences.

Koyomi has just received the three prompt strings from the contest organizers. Help her write a valid novel for the contest.

A string a is a subsequence of a string b if a can be obtained from b by deletion of several (possibly, zero) characters.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 10^5$).

Each of the following three lines contains a bitstring of length $2n$. It is guaranteed that these three strings are pairwise distinct.

It is guaranteed that the sum of n across all test cases does not exceed 10^5.

Output

For each test case, print a single line containing a bitstring of length at most $3n$ that has at least two of the given bitstrings as subsequences.

It can be proven that under the constraints of the problem, such a bitstring always exists.

If there are multiple possible answers, you may output any of them.

Example

input

```
2
1
00
11
01
3
011001
111010
010001
```

output

```
010
```

```
011001010
```

Note

In the first test case, the bitstrings `00` and `01` are subsequences of the output string: **`01`**`0` and **`0`**`1`**`0`**. Note that `11` is not a subsequence of the output string, but this is not required.

In the second test case all three input strings are subsequences of the output string: `011001010`, `011001010` and `011001010`.

E. Almost Sorted

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Seiji Maki doesn't only like to observe relationships being unfolded, he also likes to observe sequences of numbers, especially permutations. Today, he has his eyes on *almost sorted* permutations.

A permutation $a_1, a_2, \dots, a_n$ of $1, 2, \dots, n$ is said to be *almost sorted* if the condition $a_{i+1} \ge a_i - 1$ holds for all i between 1 and $n-1$ inclusive.

Maki is considering the list of all almost sorted permutations of $1, 2, \dots, n$, given in lexicographical order, and he wants to find the k-th permutation in this list. Can you help him to find such permutation?

Permutation p is lexicographically smaller than a permutation q if and only if the following holds:

- in the first position where p and q differ, the permutation p has a smaller element than the corresponding element in q.

Input

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

Each test case consists of a single line containing two integers n and k ($1 \le n \le 10^5$, $1 \le k \le 10^{18}$).

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, print a single line containing the k-th almost sorted permutation of length n in lexicographical order, or -1 if it doesn't exist.

Example

input

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

output

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

Note

For the first and second test, the list of almost sorted permutations with $n = 1$ is $\{[1]\}$.

For the third and fifth test, the list of almost sorted permutations with $n = 3$ is $\{[1, 2, 3], [1, 3, 2], [2, 1, 3], [3, 2, 1]\}$.

F. Complete the MST

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

As a teacher, Riko Hakozaki often needs to help her students with problems from various subjects. Today, she is asked a programming task which goes as follows.

You are given an undirected complete graph with n nodes, where some edges are pre-assigned with a positive weight while the rest aren't. You need to assign all unassigned edges with **non-negative weights** so that in the resulting fully-assigned complete graph the XOR sum of all weights would be equal to 0.

Define the *ugliness* of a fully-assigned complete graph the weight of its minimum spanning tree, where the weight of a spanning tree equals the sum of weights of its edges. You need to assign the weights so that the ugliness of the resulting graph is as small as possible.

As a reminder, an undirected complete graph with n nodes contains all edges (u, v) with $1 \le u < v \le n$; such a graph has $\frac{n(n-1)}{2}$ edges.

She is not sure how to solve this problem, so she asks you to solve it for her.

Input

The first line contains two integers n and m ($2 \le n \le 2 \cdot 10^5$, $0 \le m \le \min(2 \cdot 10^5, \frac{n(n-1)}{2} - 1)$) — the number of nodes and the number of pre-assigned edges. The inputs are given so that there is at least one unassigned edge.

The i-th of the following m lines contains three integers u_i, v_i, and w_i ($1 \le u_i, v_i \le n$, $u \ne v$, $1 \le w_i < 2^{30}$), representing the edge from u_i to v_i has been pre-assigned with the weight w_i. No edge appears in the input more than once.

Output

Print on one line one integer — the minimum ugliness among all weight assignments with XOR sum equal to 0.

Examples

input

```
4 4
2 1 14
1 4 14
3 2 15
4 3 8
```

output

```
15
```

input

```
6 6
3 6 4
2 4 1
4 5 7
3 4 10
3 5 1
5 2 15
```

output

```
0
```

input

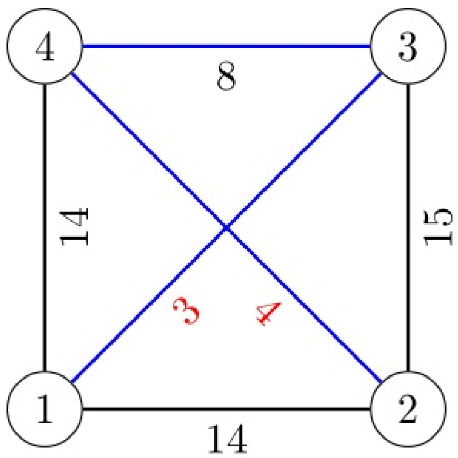
```
5 6
2 3 11
5 3 7
1 4 10
2 4 14
4 3 8
2 5 6
```

output

```
6
```

Note

The following image showcases the first test case. The black weights are pre-assigned from the statement, the red weights are assigned by us, and the minimum spanning tree is denoted by the blue edges.



Loading [MathJax]/jax/output/CommonHTML/jax.js