Codeforces Round #565 (Div. 3)

A. Divide it!

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

You are given an integer n.

You can perform any of the following operations with this number an arbitrary (possibly, zero) number of times:

1. Replace n with $\frac{n}{2}$ if n is divisible by 2;
2. Replace n with $\frac{2n}{3}$ if n is divisible by 3;
3. Replace n with $\frac{4n}{5}$ if n is divisible by 5.

For example, you can replace 30 with 15 using the first operation, with 20 using the second operation or with 24 using the third operation.

Your task is to find the minimum number of moves required to obtain 1 from n or say that it is impossible to do it.

You have to answer q independent queries.

Input
The first line of the input contains one integer q ($1 \le q \le 1000$) — the number of queries.

The next q lines contain the queries. For each query you are given the integer number n ($1 \le n \le 10^{18}$).

Output
Print the answer for each query on a new line. If it is impossible to obtain 1 from n, print -1. Otherwise, print the minimum number of moves required to do it.

Example

input
7 1 10 25 30 14 27 1000000000000000000
output
0 4 6 6 -1 6 72

B. Merge it!

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

You are given an array a consisting of n integers $a_1, a_2, \ldots, a_n$.

In one operation you can choose two elements of the array and replace them with the element equal to their sum (it does not matter where you insert the new element). For example, from the array $[2, 1, 4]$ you can obtain the following arrays: $[3, 4]$, $[1, 6]$ and $[2, 5]$.

Your task is to find the maximum possible number of elements divisible by 3 that are in the array after performing this operation an arbitrary (possibly, zero) number of times.

You have to answer t independent queries.

Input
The first line contains one integer t ($1 \le t \le 1000$) — the number of queries.

The first line of each query contains one integer n ($1 \le n \le 100$).

The second line of each query contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

Output
For each query print one integer in a single line — the maximum possible number of elements divisible by 3 that are in the array after performing described operation an arbitrary (possibly, zero) number of times.

Example

input
2 5 3 1 2 3 1 7 1 1 1 1 1 2 2
output
3 3

Note
In the first query of the example you can apply the following sequence of operations to obtain 3 elements divisible by 3: $[3, 1, 2, 3, 1] \to [3, 3, 3, 1]$.

In the second query you can obtain 3 elements divisible by 3 with the following sequence of operations: $[1, 1, 1, 1, 1, 2, 2] \to [1, 1, 1, 1, 2, 3] \to [1, 1, 1, 3, 3] \to [2, 1, 3, 3] \to [3, 3, 3]$.

C. Lose it!

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

You are given an array a consisting of n integers. Each a_i is one of the six following numbers: $4, 8, 15, 16, 23, 42$.

Your task is to remove the minimum number of elements to make this array **good**.

An array of length k is called **good** if k is divisible by 6 and it is possible to split it into $\frac{k}{6}$ **subsequences** $4, 8, 15, 16, 23, 42$.

Examples of good arrays:

- $[4, 8, 15, 16, 23, 42]$ (the whole array is a required sequence);
- $[4, 8, 4, 15, 16, 8, 23, 15, 16, 42, 23, 42]$ (the first sequence is formed from first, second, fourth, fifth, seventh and tenth elements and the second one is formed from remaining elements);
- $[]$ (**the empty array is good**).

Examples of bad arrays:

- $[4, 8, 15, 16, 42, 23]$ (the order of elements should be exactly $4, 8, 15, 16, 23, 42$);
- $[4, 8, 15, 16, 23, 42, 4]$ (the length of the array is not divisible by 6);
- $[4, 8, 15, 16, 23, 42, 4, 8, 15, 16, 23, 23]$ (the first sequence can be formed from first six elements but the remaining array cannot form the required sequence).

Input
The first line of the input contains one integer n ($1 \le n \le 5 \cdot 10^5$) — the number of elements in a.

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ (each a_i is one of the following numbers: $4, 8, 15, 16, 23, 42$), where a_i is the i-th element of a.

Output
Print one integer — the minimum number of elements you have to remove to obtain a **good** array.

Examples

input
5 4 8 15 16 23
output
5

input
12 4 8 4 15 16 8 23 15 16 42 23 42
output

0

input
15 4 8 4 8 15 16 8 16 23 15 16 4 42 23 42
output
3

D. Recover it!

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Authors guessed an array a consisting of n integers; each integer is not less than 2 and not greater than $2 \cdot 10^5$. You don't know the array a, but you know the array b which is formed from it with the following sequence of operations:

1. Firstly, let the array b be equal to the array a;
2. Secondly, for each i from 1 to n:
   - if a_i is a **prime** number, then one integer p_{a_i} is appended to array b, where p is an infinite sequence of prime numbers ($2, 3, 5, \ldots$);
   - otherwise (if a_i is not a **prime** number), the greatest divisor of a_i which is not equal to a_i is appended to b;
3. Then the obtained array of length $2n$ is shuffled and given to you in the input.

Here p_{a_i} means the a_i-th prime number. The first prime $p_1 = 2$, the second one is $p_2 = 3$, and so on.

Your task is to recover **any** suitable array a that forms the given array b. It is guaranteed that the answer exists (so the array b is obtained from some suitable array a). If there are multiple answers, you can print any.

Input

The first line of the input contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of elements in a.

The second line of the input contains $2n$ integers $b_1, b_2, \ldots, b_{2n}$ ($2 \le b_i \le 2750131$), where b_i is the i-th element of b. 2750131 is the 199999-th prime number.

Output

In the only line of the output print n integers $a_1, a_2, \ldots, a_n$ ($2 \le a_i \le 2 \cdot 10^5$) in **any** order — the array a from which the array b can be obtained using the sequence of moves given in the problem statement. If there are multiple answers, you can print any.

Examples

input
3 3 5 2 3 2 4
output
3 4 2

input
1 2750131 199999
output
199999

input
1 3 6
output
6

E. Cover it!

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

You are given an undirected unweighted connected graph consisting of n vertices and m edges. It is guaranteed that there are no self-loops or multiple edges in the given graph.

Your task is to choose **at most** $\lfloor \frac{n}{2} \rfloor$ vertices in this graph so **each** unchosen vertex is adjacent (in other words, connected by an edge) to at least one of chosen vertices.

It is guaranteed that the answer exists. If there are multiple answers, you can print any.

You will be given multiple independent queries to answer.

Input

The first line contains a single integer t ($1 \le t \le 2 \cdot 10^5$) — the number of queries.

Then t queries follow.

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

The following m lines denote edges: edge i is represented by a pair of integers v_i, u_i ($1 \le v_i, u_i \le n$, $u_i \ne v_i$), which are the indices of vertices connected by the edge.

There are no self-loops or multiple edges in the given graph, i. e. for each pair (v_i, u_i) there are no other pairs (v_i, u_i) or (u_i, v_i) in the list of edges, and for each pair (v_i, u_i) the condition $v_i \ne u_i$ is satisfied. It is guaranteed that the given graph is **connected**.

It is guaranteed that $\sum m \le 2 \cdot 10^5$ over all queries.

Output

For each query print two lines.

In the first line print k ($1 \le \lfloor \frac{n}{2} \rfloor$) — the number of chosen vertices.

In the second line print k **distinct** integers $c_1, c_2, \ldots, c_k$ in any order, where c_i is the index of the i-th chosen vertex.

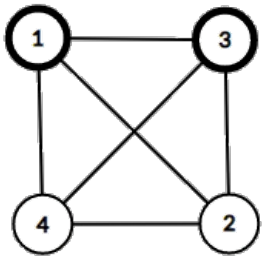
It is guaranteed that the answer exists. If there are multiple answers, you can print any.

Example

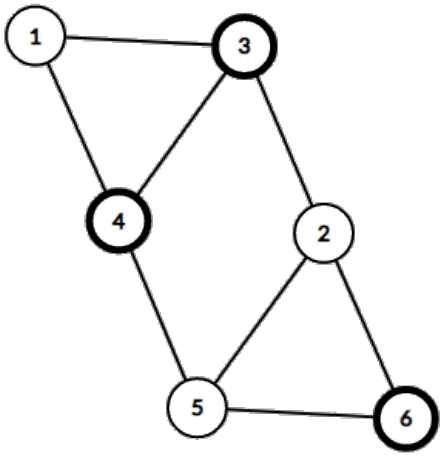
input
2 4 6 1 2 1 3 1 4 2 3 2 4 3 4 6 8 2 5 5 4 4 3 4 1 1 3 2 3 2 6 5 6
output
2 1 3 3 4 3 6

Note

In the first query any vertex or any pair of vertices will suffice.



Note that you don't have to minimize the number of chosen vertices. In the second query two vertices can be enough (vertices 2 and 4) but three is also ok.



F. Destroy it!

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

You are playing a computer card game called Splay the Sire. Currently you are struggling to defeat the final boss of the game.

The boss battle consists of n turns. During each turn, you will get several cards. Each card has two parameters: its cost c_i and damage d_i. You may play some of your cards during each turn in some sequence (you choose the cards and the exact order they are played), as long as **the total cost of the cards you play during the turn does not exceed** 3. After playing some (possibly zero) cards, you end your turn, and **all cards you didn't play are discarded**. Note that you can use each card **at most once**.

Your character has also found an artifact that boosts the damage of some of your actions: every 10-th card you play deals double damage.

What is the maximum possible damage you can deal during n turns?

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of turns.

Then n blocks of input follow, the i-th block representing the cards you get during the i-th turn.

Each block begins with a line containing one integer k_i ($1 \le k_i \le 2 \cdot 10^5$) — the number of cards you get during i-th turn. Then k_i lines follow, each containing two integers c_j and d_j ($1 \le c_j \le 3$, $1 \le d_j \le 10^9$) — the parameters of the corresponding card.

It is guaranteed that $\sum\limits_{i=1}^{n} k_i \le 2 \cdot 10^5$.

Output

Print one integer — the maximum damage you may deal.

Example

input
5 3 1 6 1 7 1 5 2 1 4 1 3 3 1 10 3 5 2 3 3 1 15 2 4 1 10 1 1 100
output
263

Note

In the example test the best course of action is as follows:

During the first turn, play all three cards in any order and deal 18 damage.

During the second turn, play both cards and deal 7 damage.

During the third turn, play the first and the third card and deal 13 damage.

During the fourth turn, play the first and the third card and deal 25 damage.

During the fifth turn, play the only card, which will deal double damage (200).
