Educational Codeforces Round 122 (Rated for Div. 2)

A. Div. 7

2 seconds, 512 megabytes

You are given an integer n. You have to change the minimum number of digits in it in such a way that the resulting number **does not have any leading zeroes** and **is divisible by** 7.

If there are multiple ways to do it, print any of them. If the given number is already divisible by 7, leave it unchanged.

Input

The first line contains one integer t ($1 \le t \le 990$) — the number of test cases.

Then the test cases follow, each test case consists of one line containing one integer n ($10 \le n \le 999$).

Output

For each test case, print one integer without any leading zeroes — the result of your changes (i. e. the integer that is divisible by 7 and can be obtained by changing the minimum possible number of digits in n).

If there are multiple ways to apply changes, print any resulting number. If the given number is already divisible by 7, just print it.

input
3 42 23 377
output
42 28 777

In the first test case of the example, 42 is already divisible by 7, so there's no need to change it.

In the second test case of the example, there are multiple answers — 28, 21 or 63.

In the third test case of the example, other possible answers are 357, 371 and 378. Note that you **cannot** print 077 or 77.

The problem statement has recently been changed. View the changes. ×

B. Minority

2 seconds, 256 megabytes

You are given a string s, consisting only of characters '0' and '1'.

You have to choose a contiguous substring of s and remove all occurrences of the character, which is a strict minority in it, from the substring.

That is, if the amount of '0's in the substring is strictly smaller than the amount of '1's, remove all occurrences of '0' from the substring. If the amount of '1's is strictly smaller than the amount of '0's, remove all occurrences of '1'. If the amounts are the same, do nothing.

You have to apply the operation **exactly once**. What is the maximum amount of characters that can be removed?

Input

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

The only line of each testcase contains a non-empty string s, consisting only of characters '0' and '1'. The length of s doesn't exceed $2 \cdot 10^5$.

The total length of strings s over all testcases doesn't exceed $2 \cdot 10^5$.

Output

For each testcase, print a single integer — the maximum amount of characters that can be removed after applying the operation **exactly once**.

input
4 01 1010101010111 00110001000 1
output
0 5 3 0

In the first testcase, you can choose substrings "0", "1" or "01". In "0" the amount of '0' is 1, the amount of '1' is 0. '1' is a strict minority, thus all occurrences of it are removed from the substring. However, since there were 0 of them, nothing changes. Same for "1". And in "01" neither of '0' or '1' is a strict minority. Thus, nothing changes. So there is no way to remove any characters.

In the second testcase, you can choose substring "10101010101". It contains 5 characters '0' and 6 characters '1'. '0' is a strict minority. Thus, you can remove all its occurrences. There exist other substrings that produce the same answer.

In the third testcase, you can choose substring "011000100". It contains 6 characters '0' and 3 characters '1'. '1' is a strict minority. Thus, you can remove all its occurrences.

C. Kill the Monster

2 seconds, 256 megabytes

Monocarp is playing a computer game. In this game, his character fights different monsters.

A fight between a character and a monster goes as follows. Suppose the character initially has health h_C and attack d_C; the monster initially has health h_M and attack d_M. The fight consists of several steps:

1. the character attacks the monster, decreasing the monster's health by d_C;
2. the monster attacks the character, decreasing the character's health by d_M;
3. the character attacks the monster, decreasing the monster's health by d_C;
4. the monster attacks the character, decreasing the character's health by d_M;
5. and so on, until the end of the fight.

The fight ends when someone's health becomes non-positive (i. e. 0 or less). If the monster's health becomes non-positive, the character wins, otherwise the monster wins.

Monocarp's character currently has health equal to h_C and attack equal to d_C. He wants to slay a monster with health equal to h_M and attack equal to d_M. Before the fight, Monocarp can spend up to k coins to upgrade his character's weapon and/or armor; each upgrade costs exactly one coin, each weapon upgrade increases the character's attack by w, and each armor upgrade increases the character's health by a.

Can Monocarp's character slay the monster if Monocarp spends coins on upgrades optimally?

Input

The first line contains one integer t ($1 \le t \le 5 \cdot 10^4$) — the number of test cases. Each test case consists of three lines:

The first line contains two integers h_C and d_C ($1 \le h_C \le 10^{15}$; $1 \le d_C \le 10^9$) — the character's health and attack;

The second line contains two integers h_M and d_M ($1 \le h_M \le 10^{15}$; $1 \le d_M \le 10^9$) — the monster's health and attack;

The third line contains three integers k, w and a ($0 \le k \le 2 \cdot 10^5$; $0 \le w \le 10^4$; $0 \le a \le 10^{10}$) — the maximum number of coins that Monocarp can spend, the amount added to the character's attack with each weapon upgrade, and the amount added to the character's health with each armor upgrade, respectively.

The sum of k over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print `YES` if it is possible to slay the monster by optimally choosing the upgrades. Otherwise, print `NO`.

input
4 25 4 9 20 1 1 10 25 4 12 20 1 1 10 100 1 45 2 0 4 10 9 2 69 2 4 2 7

output
YES NO YES YES

In the first example, Monocarp can spend one coin to upgrade weapon (damage will be equal to 5), then health during battle will change as follows: $(h_C, h_M) = (25, 9) \to (25, 4) \to (5, 4) \to (5, -1)$. The battle ended with Monocarp's victory.

In the second example, Monocarp has no way to defeat the monster.

In the third example, Monocarp has no coins, so he can't buy upgrades. However, the initial characteristics are enough for Monocarp to win.

In the fourth example, Monocarp has 4 coins. To defeat the monster, he has to spend 2 coins to upgrade weapon and 2 coins to upgrade armor.

D. Make Them Equal

2 seconds, 256 megabytes

You have an array of integers a of size n. Initially, all elements of the array are equal to 1. You can perform the following operation: choose two integers i ($1 \le i \le n$) and x ($x > 0$), and then increase the value of a_i by $\left\lfloor \frac{a_i}{x} \right\rfloor$ (i.e. make $a_i = a_i + \left\lfloor \frac{a_i}{x} \right\rfloor$).

After performing all operations, you will receive c_i coins for all such i that $a_i = b_i$.

Your task is to determine the maximum number of coins that you can receive by performing no more than k operations.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases.

The first line of each test case contains two integers n and k ($1 \le n \le 10^3$; $0 \le k \le 10^6$) — the size of the array and the maximum number of operations, respectively.

The second line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^3$).

The third line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^6$).

The sum of n over all test cases does not exceed 10^3.

Output

For each test case, print one integer — the maximum number of coins that you can get by performing no more than k operations.

input
4 4 4 1 7 5 2 2 6 5 2 3 0 3 5 2 5 4 7 5 9 5 2 5 6 3 5 9 1 9 7 6 14 11 4 6 2 8 16 43 45 9 41 15 38

output
9 0 30 167

E. Spanning Tree Queries

4 seconds, 256 megabytes

You are given a connected weighted undirected graph, consisting of n vertices and m edges.

You are asked k queries about it. Each query consists of a single integer x. For each query, you select a spanning tree in the graph. Let the weights of its edges be $w_1, w_2, \ldots, w_{n-1}$. The cost of a spanning tree is $\sum\limits_{i=1}^{n-1} |w_i - x|$ (the sum of absolute differences between the weights and x). The answer to a query is the lowest cost of a spanning tree.

The queries are given in a compressed format. The first p ($1 \le p \le k$) queries $q_1, q_2, \ldots, q_p$ are provided explicitly. For queries from $p+1$ to k, $q_j = (q_{j-1} \cdot a + b) \mod c$.

Print the xor of answers to all queries.

Input

The first line contains two integers n and m ($2 \le n \le 50$; $n - 1 \le m \le 300$) — the number of vertices and the number of edges in the graph.

Each of the next m lines contains a description of an undirected edge: three integers v, u and w ($1 \le v, u \le n$; $v \ne u$; $0 \le w \le 10^8$) — the vertices the edge connects and its weight. Note that there might be multiple edges between a pair of vertices. The edges form a connected graph.

The next line contains five integers p, k, a, b and c ($1 \le p \le 10^5$; $p \le k \le 10^7$; $0 \le a, b \le 10^8$; $1 \le c \le 10^8$) — the number of queries provided explicitly, the total number of queries and parameters to generate the queries.

The next line contains p integers $q_1, q_2, \ldots, q_p$ ($0 \le q_j < c$) — the first p queries.

Output

Print a single integer — the xor of answers to all queries.

input

```
5 8
4 1 4
3 1 0
3 5 3
2 5 4
3 4 8
4 3 4
4 2 8
5 3 9
3 11 1 1 10
0 1 2
```

output

```
4
```

input

```
6 7
2 4 0
5 4 7
2 4 0
2 1 7
2 6 1
3 4 4
1 4 8
4 10 3 3 7
3 0 2 1
```

output

```
5
```

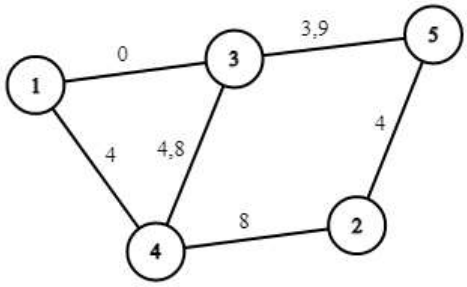
input

```
3 3
1 2 50
2 3 100
1 3 150
1 10000000 0 0 100000000
75
```

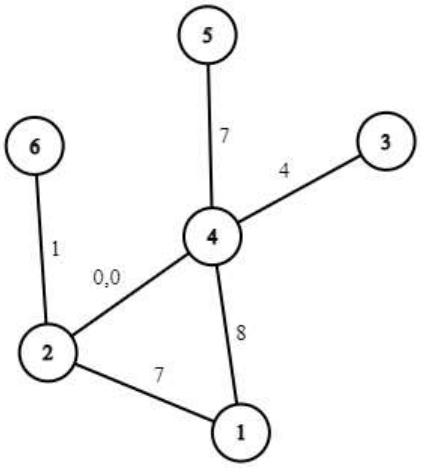
output

```
164
```

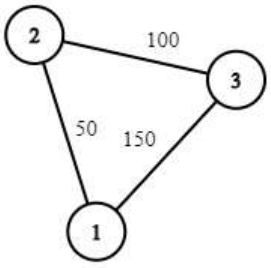
The queries in the first example are $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0$. The answers are $11, 9, 7, 3, 1, 5, 8, 7, 5, 7, 11$.



The queries in the second example are $3, 0, 2, 1, 6, 0, 3, 5, 4, 1$. The answers are $14, 19, 15, 16, 11, 19, 14, 12, 13, 16$.



The queries in the third example are $75, 0, 0, \ldots$. The answers are $50, 150, 150, \ldots$.



F. Perfect Matching

12 seconds, 512 megabytes

You are given a tree consisting of n vertices (numbered from 1 to n) and $n - 1$ edges (numbered from 1 to $n - 1$). Initially, all vertices except vertex 1 are inactive.

You have to process queries of three types:

- $1\ v$ — activate the vertex v. It is guaranteed that the vertex v is inactive before this query, and one of its neighbors is active. After activating the vertex, you have to choose a subset of edges of the tree such that each **active** vertex is incident to **exactly one** chosen edge, and each **inactive** vertex is not incident to any of the chosen edges — in other words, this subset should represent a perfect matching on the active part of the tree. If any such subset of edges exists, print the sum of indices of edges in it; otherwise, print 0.
- 2 — queries of this type will be asked only right after a query of type 1, and there will be **at most** 10 such queries. If your answer to the previous query was 0, simply print 0; otherwise, print the subset of edges for the previous query as follows: first, print the number of edges in the subset, then print the indices of the chosen edges **in ascending order**. The sum of indices should be equal to your answer to the previous query.
- 3 — terminate the program.

Note that you should solve the problem in `online` mode. It means that you can't read the whole input at once. You can read each query only after writing the answer for the last query. Use functions `fflush` in `C++` and `BufferedWriter.flush` in `Java` languages after each writing in your program.

Input

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

Then $n - 1$ lines follow. The i-th line contains two integers u_i and v_i ($1 \le u_i, v_i \le n; u_i \ne v_i$) — the endpoints of the i-th edge. These edges form a tree.

Then the queries follow in the format described in the statement, one line per query. There will be at least 2 and at most $n + 10$ queries. The last query (and only the last one) will be of type 3. Note that you can read the i-th query only if you have already given the answer for the query $i - 1$ (except for $i = 1$).

If your answer for one of the queries is incorrect and the judging program recognizes it, instead of the next query, you may receive the integer 0 on a separate line. After receiving it, your program should terminate gracefully, and you will receive "Wrong Answer" verdict. If your program doesn't terminate, your solution may receive some other verdict, like "Time Limit Exceeded", "Idleness Limit Exceeded", etc. Note that the fact that your solution doesn't receive the integer 0, it **does not mean that all your answers are correct**, some of them will be checked only after your program is terminated.

Output

For each query of type 1 or 2, print the answer on a separate line as described in the statement. Don't forget to flush the output.

input

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

output

```
1
1 1
0
0
4
2 1 3
0
0
0
```

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform