1 Check the Order

In a logistics warehouse, goods must arrive in a fixed order: 1, 2, 3, …, N. According to the company's regulations, every arriving item must first be placed on the storage rack and registered before it can be shipped out.

However, due to limited warehouse space, when a new item is placed on the rack, the existing items must be pushed deeper inside. As a result, only the outermost item can be taken out first, and items stored deeper can only be retrieved after the ones in front of them are removed.

Now, the logistics company wants to load the goods according to a specific "shipping order." You must determine whether this shipping order can be achieved under the above rules.

Input

The first line of input gives the number of test cases, T ($1 \leq T \leq 50$). Then T test cases follow each described in the following way. The first line of each test case contains the integer N described above. For each of the next lines of the test case there is a permutation of 1, 2, … , N.

Output

The output contains the lines corresponding to the lines with permutations in the input. A line of the output contains 'Yes' if it is possible to ship out goods in the order required on the corresponding line of the input. Otherwise, it contains 'No'.

Sample Input

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

Sample Output

```
Yes
No
```

2 Maximize the Energy

In the futuristic Galactic Racing League, each racer's spacecraft is equipped with a large number of energy chips, whose energy values are recorded as a continuous sequence of digits. However, due to a malfunction in the race control center's transmission system, all the chip energy values have been concatenated into one long string without any separators.

As a technician, you must devise the optimal strategy for distributing the chip energies by splitting this digit string into multiple valid energy chip values, subject to the following rules:

1. Each chip energy value after splitting must not contain leading zeros. However, the single-digit value 0 itself is considered valid.

2. Each chip energy value must fall within the range of a 32-bit signed integer.

Your goal is to determine a way of splitting the string so that the total sum of all chip energies is maximized, thereby granting the racer the greatest possible acceleration and advantage in the competition.

Input

The input begins with an integer T (≤ 500) which indicates the number of test cases followed. Each of the following test cases consists of a string of at most 200 digits.

Output

For each input, print out required answer in a single line.

Sample Input

```
6
1234554321
5432112345
000
121212121212
2147483648
11111111111111111111111111111111111111111111111111
```

Sample Output

```
1234554321
543211239
0
2121212124
214748372
5555555666
```

3 Minimum Pushes to Exit the Ruin

In the distant future, within an abandoned interstellar ruin, you are a pioneer (represented by 2) who has been stranded because your spaceship has run out of energy. During your exploration, you discover a massive object that can serve as your ship's energy core. To restart the spaceship and escape, you must push this heavy energy core (represented by 3) to the location of your spaceship (represented by 4). The ruin is structured as a vast m × n grid, where each cell may be:

1. Solid ground (0): passable terrain,
2. Collapsed zone (1): completely impassable.

You must proceed carefully, since you cannot walk through the energy core, nor can you step onto collapsed zones. Your movement is limited to the four cardinal directions: upward, downward, leftward, or rightward.

To push the energy core, you must stand on an adjacent square next to it and

move in the core's direction. Each push consumes a huge amount of energy, so your goal is to minimize the number of pushes required to move the core to the spaceship.

Your task: Calculate the minimum number of pushes required to move the energy core to the spaceship. If it is impossible, output -1.

Notes:

- Solid ground = 0
- Collapsed zone = 1
- Your starting position = 2
- Energy core = 3
- Spaceship location = 4

You only need to count the number of pushes, not the actual path taken. You cannot walk through the core or collapsed zones, but you may walk over the spaceship's location.

Input

The first line of input gives the number of test cases, T ($1 \le T \le 50$). Then T test cases follow each described in the following way. The first line of each test case contains m ($1 \le m \le 20$) and n ($1 \le n \le 20$) separated by a space, which indicates that ruin is a m × n matrix. Then m lines follow each of the n containing row i. Row contains exactly n elements integer separated by a space. j-th number in row i is the element ruin[i][j] of matrix you have to process.

Output

For each input produce one line of output. This line contains an integer which indicates the number of minimum number of pushes to move the box to the target. If there is no way to reach the location of spaceship, return -1.

Sample Input

```
3
6 6
1 1 1 1 1 1
1 4 1 1 1 1
1 0 0 3 0 1
1 0 1 1 0 1
```

```
1 0 0 0 2 1
1 1 1 1 1 1
6 6
1 1 1 1 1 1
1 4 1 1 1 1
1 0 0 3 0 1
1 1 1 1 0 1
1 0 0 0 2 1
1 1 1 1 1 1
6 6
1 1 1 1 1 1
1 4 0 0 1 1
1 0 1 3 0 1
1 0 0 0 0 1
1 0 0 0 2 1
1 1 1 1 1 1
```

Sample Output

```
3
-1
5
```

4 Check the Order

Given a number **N**, determine whether a given permutation of the numbers **1, 2, ..., N** is a valid **stack permutation**.

- This question is the same as the previous assignment.
- You must use the following structure, or else you will receive a score of 0.
- You shouldn't modify the fields in Node or Stack. However, you can modify the values if you want.

```
typedef struct Node {
    // you can add data type you want
    struct Node *next;
} Node;

typedef struct {
    // you can add data type you want
    Node *top;
} Stack;
```

Input

The first line of input gives the number of test cases, T (1≤T≤50). Then T test cases follow each described in the following way. The first line of each test case contains the integer N described above. For each of the next lines of the test case there is a permutation of 1, 2, …, N.

Output

The output contains the lines corresponding to the lines with permutations in the input. A line of the output contains ‘Yes’ if it is possible to ship out goods in the order required on the corresponding line of the input. Otherwise, it contains ‘No’.

Sample Input

2
5
1 2 3 4 5
5
5 4 1 2 3

Sample Output

Yes
No

5 Discard the Number

Given a sequence $\{x_1, x_2, \ldots, x_m\}$, where all numbers in the sequence are positive integers. Perform the following operation on this sequence: start by looking at x_1, then x_2, and continue in this manner. After reaching x_m, start again from x_1, repeating this process cyclically. Each time you look at a number, say x_i, remember it and compare it with the last remembered number x_j. If the two numbers satisfy $\gcd(x_i, x_j) = 1$, discard x_i from the sequence and forget it. Then, restart the process from x_{i+1}. This means that after discarding a number, you cannot immediately discard the next number in the subsequent step.

Your task is to determine which numbers will be discarded and output them in the order they are discarded.

- You need to represent the elements as a doubly circular linked list using the following structure, or else you will receive a score of 0.
- You shouldn't modify the fields in Node or LinkedList. However, you can modify the values if you want.
- Traversing the number and constructing it with an array is not allowed.

```
struct Node {
    int order, num;
    struct Node *next, *prev;
};

struct LinkedList {
    struct Node *head;
};
```

Input

The first line of input gives the number of testcases, T($1 \le T \le 50$). Then T test cases follow each described in the following way. The first line contains a single integer m ($1 \le m \le 5 \times 10^4$) indicates the number of elements. The second line contains the integers $x_1, x_2, \ldots, x_m$

($1 \leq x_i \leq 10^9$) separated by spaces, which indicate the number on elements.

Output

For each input produce one line of output. First, print a single integer k, which indicates the number of discarded elements. After that print k distinct integer: discarded elements in the order of their deletion.

Sample Input

```
5
5
5 9 2 10 15
6
1 2 4 2 4 2
2
1 2
1
1
1
2
```

Sample Output

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

6 Determine Base

Blackbeard, a ruthless and notorious pirate, carries out raids across the seas in pursuit of his own ambitions. As his trusted confidant, you are responsible for his calculations and strategies.

One day, he acquires a nautical chart of an archipelago, and he decides to make these islands his next target of conquest. He needs to choose one island from the archipelago as his base of operations, and it is

your task to determine which island is the most suitable, based on the information from the chart.

First, you assign a number to each island to establish how many islands exist on the map, with numbering starting from 1. According to the chart, some routes between islands are too dangerous for sailing, so only the marked safe routes can be used to travel. You may freely move back and forth between islands connected by these safe routes. You also discover that the sailing time between any two connected islands is the same.

Therefore, you decide to classify the islands into two categories: strategic hubs and ordinary islands. A strategic hub is defined as an island that appears on more than one route, and there may be multiple such hubs within the archipelago. From these hubs, you must select one as Blackbeard's base.

The criterion for choosing the base is as follows:

- The average sailing time from the base to all other strategic hubs must be minimized.

• If you use queue in this question, you must use the following structure, or else you will receive a score of 0.

• You can add data type in Node or Queue.

```c
typedef struct Node {
    // you can add data type you want
    struct Node *next, *pre;
} Node;

typedef struct Queue {
    // you can add data type you want
    Node *front, *rear;
} Queue;
```

Input

The first line indicates the number of test cases. For each test case, the first line contains two integers: N and S. N is the total number of islands, and S is the number of routes. Islands are numbered from 1 to N; N is not greater than 10000; and S is not greater than 100. Next, we have S lines, one for each. These lines consist of a list of islands, separated by blank spaces, ending with a '0'. There will be between 1 and 100 hubs, inclusive. There is always a path between any pair of islands.

Output

For each test case, you have to output the number of the resulting hub. If there are more than one hub with the minimum distance, then you have to output the one with the smallest number.

Sample Input

```
4
13 3
1 2 3 4 5 6 7 0
8 9 4 10 13 0
11 2 12 9 6 7 0
6 2
2 5 3 6 1 4 0
4 1 6 3 5 2 0
5 2
1 2 3 4 5 0
3 5 1 4 2 0
7 2
3 5 1 2 4 7 6 0
3 6 1 0
```

Sample Output

```
9
3
4
6
```

7 Construct binary tree from inorder and postorder traversal

Given two integer arrays inorder and postorder where inorder is the

inorder traversal of a binary tree and postorder is the postorder traversal of the same tree, print the preorder traversal of the binary tree.

Input

The first line of input gives the number of test cases, T ($1 \le T \le 300$). Then T test cases follow each described in the following way:

1. The first line contains a single integer n ($1 \le n \le 5 \times 10^5$) indicates the length of array inorder (inorder and postorder have the same length).
2. The second line contains n integers $in_1, in_2, ..., in_n$ ($1 \le in_i \le 6 \times 10^5$) separated by spaces, which indicate the inorder traversal of the binary tree.
3. The third line contains n integers $post_1, post_2, ..., post_n$ ($1 \le post_i \le 6 \times 10^5$) separated by spaces, which indicate the postorder traversal of the binary tree.

note: inorder and postorder consist of unique values.

Output

For each input produce one line of output. This line contains the preorder of the binary tree (separated by spaces).

Sample Input

```
2
5
9 3 15 20 7
9 15 7 20 3
7
1 2 3 4 5 6 7
1 3 2 5 7 6 4
```

Sample Output

```
3 9 20 15 7
4 2 1 3 6 5 7
```

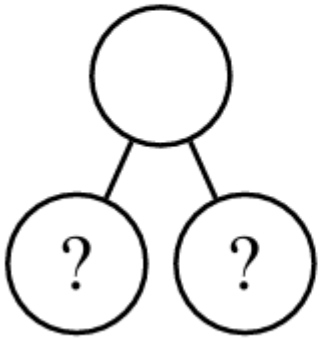
8 Set base stations

In city T, there is a binary-tree–structured network. A telecommunications company plans to install new base stations in the city. When a base station is placed on a node of the tree, its signal can only cover the adjacent nodes—that is, the node's parent and its children.

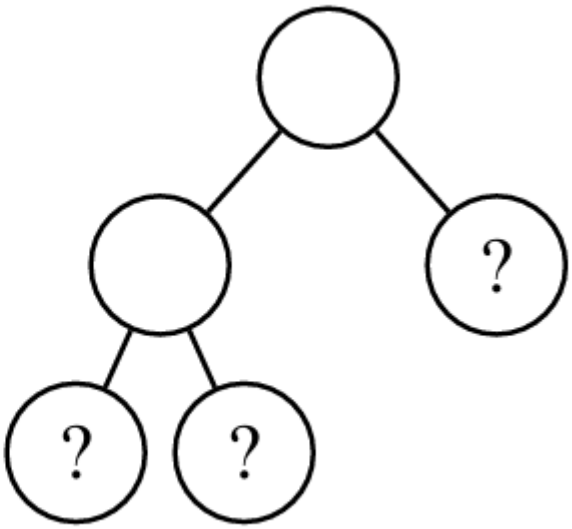
Now, please compute the minimum number of base stations the company needs to install in order to ensure that every node in the city receives signal coverage.

Input

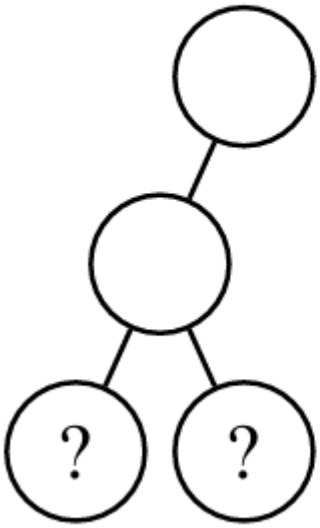
The first line of input gives the number of test cases, T ($1 \leq T \leq 300$). Then T test cases follow each described in the following way. Each line represents a test case, composed of 0 and null, separated by commas. The number of nodes in the tree is in the range [1, 5×10^5]. Nodes are filled from the upper level to the lower level, from left to right. The following is an example of input [0,0,N,0,0], N represents NULL:



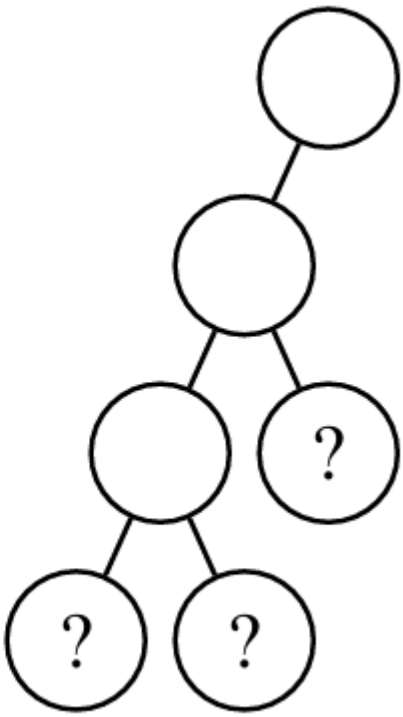
[0,0,N,0,0]



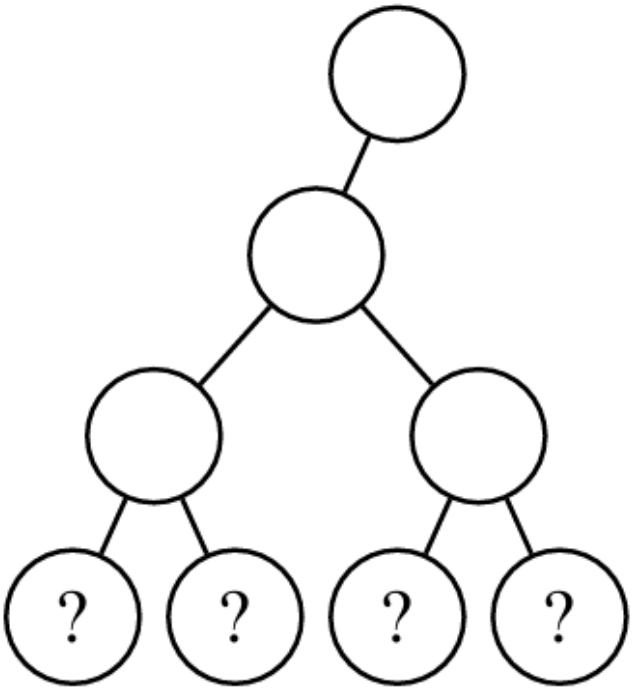
[0,0,N,0,0]



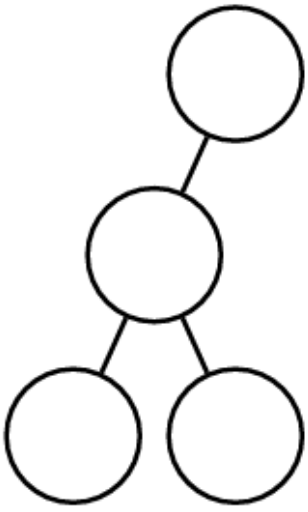
[0,0,N,0,0]



[0,0,N,0,0]



[0,0,N,0,0]



Final binary tree

Output

For each input, produce one line of output. This line contains an integer

that indicates the minimum number of base stations the company needs to install.

Sample Input

2
0,0,N,0,0
0,0,N,0,N,0,N,N,0

Sample Output

1
2

9 The Champagne Tower

At a banquet, a champagne tower is set up, represented as a binary tree. Guests will perform actions according to the host's instructions:

1. Operation 1 (format: 1 a): Pour champagne starting from node a. This will cause a and all of its descendants to become filled with champagne.
2. Operation 2 (format: 2 b): Sip champagne starting from node b. This will cause b and all of its ancestors to have their champagne drained.
3. Operation 3 (format: 3 c): Output whether node c is currently filled with champagne. Output 1 if it is filled, otherwise output 0.

Input

The first line of input gives the number of test cases, T ($1 \leq T \leq 300$). Then T test cases follow each described in the following way:

1. The first line contains a single integer n ($1 \leq n \leq 5 \times 10^5$) indicates the length of array inorder (inorder and preorder have the same length).
2. The second line contains n integers $in_1, in_2, ..., in_n$ ($1 \leq in_i \leq n$) separated by spaces, which indicate the the inorder traversal of the binary tree.
3. The third line contains n integers $pre_1, pre_2, ..., pre_n$ ($1 \leq pre_i$

≤ n) separated by spaces, which indicate the the preorder traversal of the binary tree.

4. The fourth line contains includes a number q ($1 \le q \le 5 \times 10^5$), which represents the quantity of operations to be executed.

5. In each of the subsequent q lines, you will find two numbers, c_i ($1 \le c_i \le 3$) and v_i ($1 \le v_i \le$n), separated by spaces. Here, c_i represents the type of operation, and v_i signifies the node on which the operation is carried out.

Type of operation:

1. Fill node a with champagne. Then a and all its descendants are filled with champagne.
2. Sip node b. Then node b and all its ancestors are drained.
3. Determine whether node c is filled with champagne at the moment.

note: inorder and preorder consist of unique values.

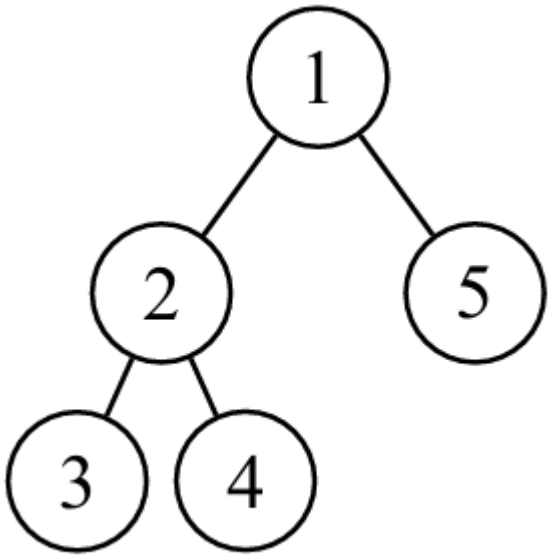
Output

For every type 3 operation, output 1 on a new line if the node n is currently filled with champagne, and 0 if the node is not filled with champagne at that moment. Ensure that the outputs for the type 3 operations are presented in the same sequence as they appear in the input.

Sample Input

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

3 5



Input

Sample Output

0
0
0
1
0
1
0
1