Computer Based Test

1. *For each question, there will be a testcase (e.g. the testcase for question 1 is* testcase1.txt*) Please note that the testcase you received may not necessarily to be the same as the testcase used for grading. Please consider edge cases.*

2. *Let your program read the corresponding testcase and print the result to a corresponding text file(e.g. the output for question 2 should be* output2.txt*).*

3. *For each question, we have provided a corresponding answer (e.g assume you're solving problem 3, the corresponding answer is* ans3.txt*), and your output should be same as answer.*

4. *For each question, your program file should be named* DS{problem number}_{student ID}.c. *(e.g. assume your student ID is* 7112056067 *and you're solving question 3, your program file should be* DS3_7112056067.c)

5. *Each question is scored independently, but partial credit is not awarded; full credit is given only for complete correctness.*

6. *You can submit for three times.*

1 Problem 1

The city is connected by n intersections with $n-1$ two-way roads, which happen to form an uninterrupted transportation network. Each road has a construction cost, and the government wants to gradually replace existing roads to reduce the overall capital cost. Lately there have been traffic jams at several points, almost at all times.

Experts recommend a simple solution: just change some two-way blocks to be one-way blocks. However, it is clear that this should be done carefully, because accessibility among city points may be lost. On the other hand, even if accessibility is guaranteed, it is possible that distances between specific intersections may be significantly augmented. After a lot of discussions, the Mayor's advisers have recommended to accept any proposal that increases the distance between any two intersections by a factor A augmented by a constant B, with respect to the old configuration (i.e., if the actual distance from one intersection to another is x, then the new distance should be at most A · x+B).

Hint: BFS / Floyd-Warshall algorithm.

Input Format

The input consists of multiple test cases:

- Each test case begins with a line containing two integers n and m ($1 \leq n \leq 10^4, m = n-1$), indicating the number of intersections and the number of roads (which always form a tree).
- Next, m lines follow, each with three integers: u, v, c, describing a road connecting intersections u and v with cost c.
- Then, an integer q follows —the number of replacement proposals.
- Each of the next q lines contains a proposal in the form of three integers u, v, c', indicating that the road connecting u and v may be replaced with a new road of cost c'.

Output Format

For each proposal:

- Output `YES` and the new total cost if the replacement is accepted.
- Output `NO` otherwise.

Sample Input

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

```
2 5
3 1 4
4 5
5 3
1 2
5
1 2 3
2 1 5
3 4 5 1
4 3 5
5 2 3 4
1 2
2 5
3 1 4
4 5
5 3
2 0
3
1 2
2 1 3
3 1 2
1 2
2 3
3 1
0 2
0
```

Sample Output

```
Yes 2
No 2
Yes 2
```

2 Problem 2

You find yourself trapped inside a secret underground research facility, composed of multiple levels of rooms and corridors. The facility is structured as a 3D grid with H floors, each floor containing R rows and C columns of rooms.

Rules

To complete this task, you must implement a breadth-first search (BFS) using a manually constructed queue based on a singly linked list.

You are **not allowed to use standard library containers** such as `queue`, `deque`, `list`, `vector`, etc.

- You must use the following structure, or else you will receive a score of 0.

```
struct Node {
int z, x, y;              // Coordinate
char val;                 //('S', 'E', '.', '#')
bool visited;
Node* neighbors[6];
Node* next;               // for BFS queue
};
```

- And then use a queue to perform the BFS, you must implement the following operations:
- `enqueue()`
- `dequeue();`
- `bool isEmpty();`

Input Format

Each test case begins with three integers H, R, and C ($1 \leq H, R, C \leq 30$), the number of levels, rows, and columns, respectively.

Then H blocks follow, each describing a single floor with R lines of C characters each. These characters represent:

- `S` —your starting position
- `E` —the exit
- `.` —open space (can move into)
- `#` —blocked room (impassable)

There is one empty line separating each floor.

The input ends when a line containing `0 0 0` is encountered.

Output Format

For each test case, output one of the following:
$Escaped\ in\ X\ minute(s)$ or $Trapped$!
Where `X` is the minimum number of minutes required to escape.

Sample Input

```
3 4 5
S....
.###.
.##..
###.#

#####
#####
##.##
##...

#####
#####
#.###
####E

1 3 3
S##
#E#
###

0 0 0
```

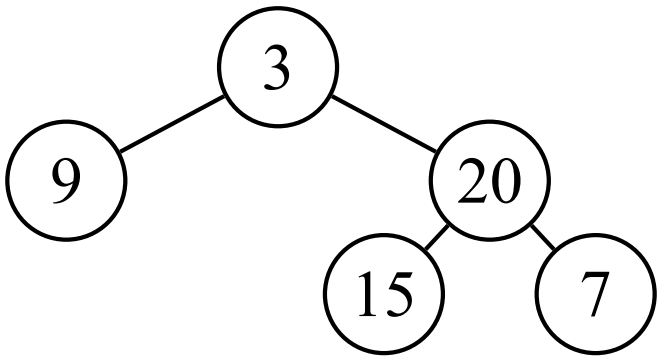
Sample Output

```
Escaped in 11 step(s).
Trapped!
```

3 Problem 3

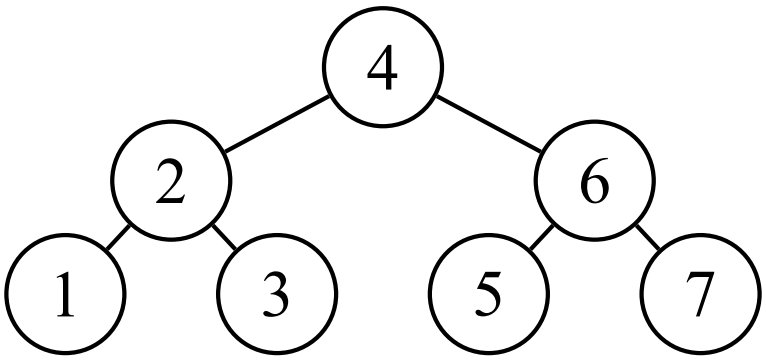
Given two integer arrays **inorder** and **postorder** where inorder is the inorder traversal of a binary tree and postorder is the postorder traversal of the same tree, print the **preorder** traversal of the binary tree.

Example 1:



- $inorder = [9, 3, 15, 20, 7]$
- $postorder = [9, 15, 7, 20, 3]$
- $output = [3, 9, 20, 15, 7]$

Example 2:



- $inorder = [1, 2, 3, 4, 5, 6, 7]$
- $postorder = [1, 3, 2, 5, 7, 6, 4]$
- $output = [4, 2, 1, 3, 6, 5, 7]$

Input Format

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

1. The first line contains a single integer $\boldsymbol{n}$ $(\mathbf{1} \leq \boldsymbol{n} \leq \mathbf{5} \times \mathbf{10^5})$ indicates the length of array **inorder (inorder and postorder have the same length)**.
2. The second line contains $\boldsymbol{n}$ integers $\boldsymbol{in_1, in_2, ..., in_n}$ $(\mathbf{1} \leq \boldsymbol{in_i} \leq \mathbf{6} \times \mathbf{10^5})$ separated by spaces, which indicate the the inorder traversal of the binary tree.

3. The third line contains n integers $post_1, post_2, ..., post_n$ $(1 \leq post_i \leq 6 \times 10^5)$ separated by spaces, which indicate the the postorder traversal of the binary tree.

note: inorder and postorder consist of unique values.

Output Format

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
```