Codeforces Round 944 (Div. 4)

A. My First Sorting Problem

1 second, 256 megabytes

You are given two integers x and y.

Output two integers: the minimum of x and y, followed by the maximum of x and y.

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

The only line of each test case contains two space-separated integers x and y ($0 \le x, y \le 9$).

Output
For each test case, output two integers: the minimum of x and y, followed by the maximum of x and y.

input

```
10
1 9
8 4
1 4
3 4
2 0
2 4
6 9
3 3
0 0
9 9
```

output

```
1 9
4 8
1 4
3 4
0 2
2 4
6 9
3 3
0 0
9 9
```

B. Different String

1 second, 256 megabytes

You are given a string s consisting of lowercase English letters.

Rearrange the characters of s to form a new string r that is **not equal** to s, or report that it's impossible.

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

The only line of each test case contains a string s of length at most 10 consisting of lowercase English letters.

Output
For each test case, if no such string r exists as described in the statement, output "NO" (without quotes).

Otherwise, output "YES" (without quotes). Then, output one line — the string r, consisting of letters of string s.

You can output "YES" and "NO" in any case (for example, strings "yEs", "yes", and "Yes" will be recognized as a positive response).

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

input

```
8
codeforces
aaaaa
xxxxy
co
d
nutdealer
mwistht
hhhhhhhhhh
```

output

```
YES
forcodesec
NO
YES
xxyxx
YES
oc
NO
YES
undertale
YES
thtsiwm
NO
```

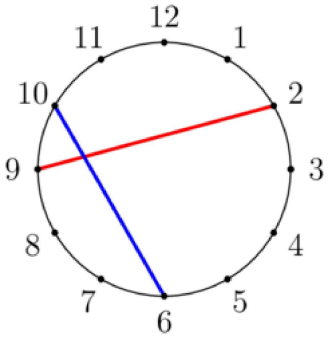
In the first test case, another possible answer is `forcescode`.

In the second test case, all rearrangements of `aaaaa` are equal to `aaaaa`.

C. Clock and Strings

1 second, 256 megabytes

There is a clock labeled with the numbers 1 through 12 in clockwise order, as shown below.



In this example, $(a, b, c, d) = (2, 9, 10, 6)$, and the strings intersect.

Alice and Bob have four **distinct** integers a, b, c, d not more than 12. Alice ties a red string connecting a and b, and Bob ties a blue string connecting c and d. Do the strings intersect? (The strings are straight line segments.)

Input
The first line contains a single integer t ($1 \le t \le 5940$) — the number of test cases.

The only line of each test case contains four **distinct** integers a, b, c, d ($1 \le a, b, c, d \le 12$).

Output
For each test case, output "YES" (without quotes) if the strings intersect, and "NO" (without quotes) otherwise.

You can output "YES" and "NO" in any case (for example, strings "yEs", "yes", and "Yes" will be recognized as a positive response).

input

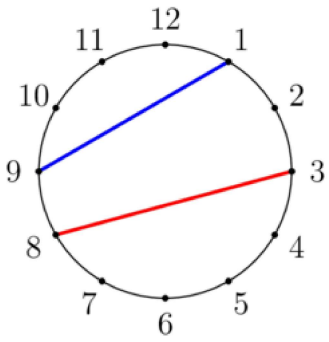
```
15
2 9 10 6
3 8 9 1
1 2 3 4
5 3 4 12
1 8 2 10
3 12 11 8
9 10 12 1
12 1 10 2
3 12 6 9
1 9 8 4
6 7 9 12
7 12 9 6
10 12 11 1
3 9 6 12
1 4 3 5
```

output

```
YES
NO
NO
YES
YES
NO
NO
NO
NO
NO
NO
YES
YES
YES
YES
```

The first test case is pictured in the statement.

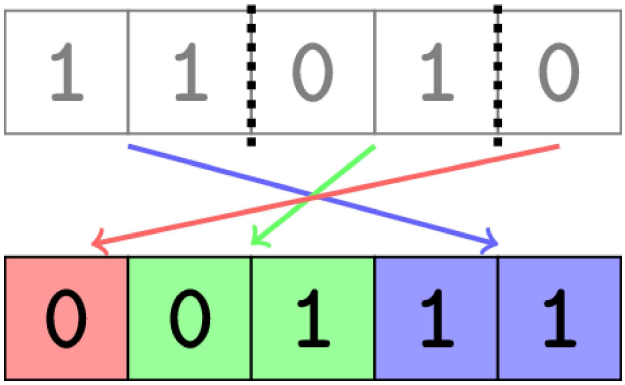
In the second test case, the strings do not intersect, as shown below.



D. Binary Cut

2 seconds, 256 megabytes

You are given a binary string[†] . Please find the minimum number of pieces you need to cut it into, so that the resulting pieces can be rearranged into a sorted binary string.



Note that:

- each character must lie in exactly one of the pieces;
- the pieces must be contiguous substrings of the original string;
- you must use all the pieces in the rearrangement.

[†] A *binary string* is a string consisting of characters `0` and `1`. A *sorted binary string* is a binary string such that all characters `0` come before all characters `1`.

Input

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

The only line of each test case contains a single string s ($1 \le |s| \le 500$) consisting of characters `0` and `1`, where $|s|$ denotes the length of the string s.

Output

For each test case, output a single integer — the minimum number of pieces needed to be able to rearrange the string into a sorted binary string.

input

```
6
11010
00000000
1
10
0001111
0110
```

output

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

The first test case is pictured in the statement. It can be proven that you can't use fewer than 3 pieces.

In the second and third test cases, the binary string is already sorted, so only 1 piece is needed.

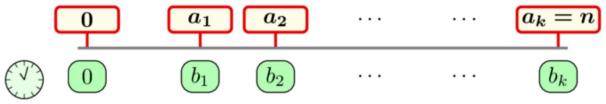
In the fourth test case, you need to make a single cut between the two characters and rearrange them to make the string `01`.

E. Find the Car

3 seconds, 256 megabytes

Timur is in a car traveling on the number line from point 0 to point n. The car starts moving from point 0 at minute 0.

There are $k + 1$ signs on the line at points $0, a_1, a_2, \ldots, a_k$, and Timur knows that the car will arrive there at minutes $0, b_1, b_2, \ldots, b_k$, respectively. The sequences a and b are strictly increasing with $a_k = n$.



Between any two adjacent signs, the car travels with a **constant speed**. Timur has q queries: each query will be an integer d, and Timur wants you to output how many minutes it takes the car to reach point d, **rounded down to the nearest integer**.

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 three integers n, k, and q, ($k \le n \le 10^9$; $1 \le k, q \le 10^5$) — the final destination, the number of points Timur knows the time for, and the number of queries respectively.

The second line of each test case contains k integers a_i ($1 \le a_i \le n$; $a_i < a_{i+1}$ for every $1 \le i \le k-1$; $a_k = n$).

The third line of each test case contains k integers b_i ($1 \le b_i \le 10^9$; $b_i < b_{i+1}$ for every $1 \le i \le k-1$).

Each of the following q lines contains a single integer d ($0 \le d \le n$) — the distance that Timur asks the minutes passed for.

The sum of k over all test cases doesn't exceed 10^5, and the sum of q over all test cases doesn't exceed 10^5.

Output

For each query, output a single integer — the number of minutes passed until the car reaches the point d, rounded down.

input

```
4
10 1 3
10
10
0
6
7
10 2 4
4 10
4 7
6
4
2
7
1000000000 1 1
1000000000
1000000000
99999999
6 1 3
6
5
2
6
5
```

output

```
0 6 7
5 4 2 5
99999999
1 5 4
```

For the first test case, the car goes from point 0 to point 10 in 10 minutes, so the speed is 1 unit per minute and:

- At point 0, the time will be 0 minutes.
- At point 6, the time will be 6 minutes.
- At point 7, the time will be 7 minutes.

For the second test case, between points 0 and 4, the car travels at a speed of 1 unit per minute and between 4 and 10 with a speed of 2 units per minute and:

- At point 6, the time will be 5 minutes.
- At point 4, the time will be 4 minutes.
- At point 2, the time will be 2 minutes.
- At point 7, the time will be 5.5 minutes, so the answer is 5.

For the fourth test case, the car travels with 1.2 units per minute, so the answers to the queries are:

- At point 2, the time will be $1.66\ldots$ minutes, so the answer is 1.
- At point 6, the time will be 5 minutes.
- At point 5, the time will be $4.16\ldots$ minutes, so the answer is 4.

F. Circle Perimeter

1 second, 256 megabytes

Given an integer r, find the number of lattice points that have a Euclidean distance from $(0, 0)$ **greater than or equal to** r but **strictly less** than $r + 1$.

A *lattice point* is a point with integer coordinates. The *Euclidean distance* from $(0, 0)$ to the point (x, y) is $\sqrt{x^2 + y^2}$.

Input

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

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

The sum of r over all test cases does not exceed 10^5.

Output

For each test case, output a single integer — the number of lattice points that have an Euclidean distance d from $(0, 0)$ such that $r \le d < r + 1$.

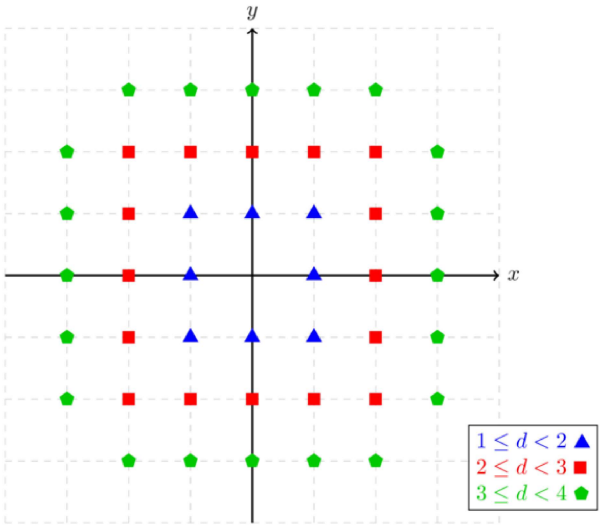
input

```
6
1
2
3
4
5
1984
```

output

```
8
16
20
24
40
12504
```

The points for the first three test cases are shown below.



G. XOUR

2 seconds, 256 megabytes

You are given an array a consisting of n nonnegative integers.

You can swap the elements at positions i and j if a_i XOR $a_j < 4$, where XOR is the bitwise XOR operation.

Find the lexicographically smallest array that can be made with any number of swaps.

An array x is lexicographically smaller than an array y if in the first position where x and y differ, $x_i < y_i$.

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 ($1 \le n \le 2 \cdot 10^5$) — the length of the array.

The second line of each test case contains n integers a_i ($0 \le a_i \le 10^9$) — the elements of the array.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output n integers — the lexicographically smallest array that can be made with any number of swaps.

input

```
4
4
1 0 3 2
5
2 7 1 5 6
8
1 2 1 2 1 2 1 2
4
16 4 1 64
```

output

```
0 1 2 3
1 5 2 6 7
1 1 1 1 2 2 2 2
16 4 1 64
```

For the first test case, you can swap any two elements, so we can produce the sorted array.

For the second test case, you can swap 2 and 1 (their XOR is 3), 7 and 5 (their XOR is 2), and 7 and 6 (their XOR is 1) to get the lexicographically smallest array.

H. ±1

2 seconds, 256 megabytes

Bob has a grid with 3 rows and n columns, each of which contains either a_i or $-a_i$ for some integer $1 \le i \le n$. For example, one possible grid for $n = 4$ is shown below:

$$\begin{bmatrix} a_1 & -a_2 & -a_3 & -a_2 \\ -a_4 & a_4 & -a_1 & -a_3 \\ a_1 & a_2 & -a_2 & a_4 \end{bmatrix}$$

Alice and Bob play a game as follows:

- Bob shows Alice his grid.
- Alice gives Bob an array $a_1, a_2, \ldots, a_n$ of her choosing, **whose elements are all -1 or 1**.
- Bob substitutes these values into his grid to make a grid of -1s and 1s.
- Bob **sorts** the elements of each column in non-decreasing order.
- Alice wins if all the elements in the middle row are 1; otherwise, Bob wins.

For example, suppose Alice gives Bob the array $[1, -1, -1, 1]$ for the grid above. Then the following will happen (colors are added for clarity):

$$\begin{bmatrix} a_1 & -a_2 & -a_3 & -a_2 \\ -a_4 & a_4 & -a_1 & -a_3 \\ a_1 & a_2 & -a_2 & a_4 \end{bmatrix} \xrightarrow{[1,-1,-1,1]} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix} \xrightarrow{\text{sort each column}} \begin{bmatrix} -1 & -1 & -1 & 1 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Since the middle row is all 1, Alice wins.

Given Bob's grid, determine whether or not Alice can choose the array a to win the game.

Input

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

The first line of each test case contains a single integer n ($2 \le n \le 500$) — the number of columns of Bob's grid.

The next three lines each contain n integers, the i-th of which contains $g_{i,1}, g_{i,2}, \ldots, g_{i,n}$ ($-n \le g_{i,j} \le n$, $g_{i,j} \neq 0$), representing Bob's grid.

If cell $x > 0$ is in the input, that cell in Bob's grid should contain a_x; if $x < 0$ is in the input, that cell in Bob's grid should contain $-a_{-x}$. See the sample input and notes for a better understanding.

Output

For each test case, output "YES" (without quotes) if Alice can win, and "NO" (without quotes) otherwise.

You can output "YES" and "NO" in any case (for example, strings "yEs", "yes", and "Yes" will be recognized as a positive response).

input
4 4 1 -2 -3 -2 -4 4 -1 -3 1 2 -2 4 2 1 2 -1 -2 2 -2 5 1 2 3 4 5 -2 3 -4 -5 -1 3 -5 1 2 2 6 1 3 -6 2 5 2 1 3 -2 -3 -6 -5 -2 -1 -3 2 3 1
output
YES NO YES NO

The first test case is described in the statement.

In the second test case, Bob's grid is as follows:

$$\begin{bmatrix} a_1 & a_2 \\ -a_1 & -a_2 \\ a_2 & -a_2 \end{bmatrix}$$

For the last column to have 1 in the middle row when sorted, Alice must pick $a_2 = -1$. However, it is then impossible to choose a_1 such that the first column has 1 in the middle when sorted. Thus, Alice cannot win.

In the third test case, Alice can pick $a = [1, 1, 1, 1, 1]$.