Educational Codeforces Round 119 (Rated for Div. 2)

A. Equal or Not Equal

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

You had n positive integers $a_1, a_2, \dots, a_n$ arranged *in a circle*. For each pair of neighboring numbers (a_1 and a_2, a_2 and a_3, ..., a_{n-1} and a_n, and a_n and a_1), you wrote down: are the numbers in the pair equal or not.

Unfortunately, you've lost a piece of paper with the array a. Moreover, you are afraid that even information about equality of neighboring elements may be inconsistent. So, you are wondering: is there any array a which is consistent with information you have about equality or non-equality of corresponding pairs?

Input

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

The first and only line of each test case contains a non-empty string s consisting of characters E and/or N. The length of s is equal to the size of array n and $2 \le n \le 50$. For each i from 1 to n:

- if $s_i =$ E then a_i is equal to a_{i+1} ($a_n = a_1$ for $i = n$);
- if $s_i =$ N then a_i is not equal to a_{i+1} ($a_n \neq a_1$ for $i = n$).

Output

For each test case, print YES if it's possible to choose array a that are consistent with information from s you know. Otherwise, print NO.

It can be proved, that if there exists some array a, then there exists an array a of positive integers with values less or equal to 10^9.

Example

input

```
4
EEE
EN
ENNEENE
NENN
```

output

```
YES
NO
YES
YES
```

Note

In the first test case, you can choose, for example, $a_1 = a_2 = a_3 = 5$.

In the second test case, there is no array a, since, according to s_1, a_1 is equal to a_2, but, according to s_2, a_2 is not equal to a_1.

In the third test case, you can, for example, choose array $a = [20, 20, 4, 50, 50, 50, 20]$.

In the fourth test case, you can, for example, choose $a = [1, 3, 3, 7]$.

B. Triangles on a Rectangle

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

A rectangle with its opposite corners in $(0, 0)$ and (w, h) and sides parallel to the axes is drawn on a plane.

You are given a list of lattice points such that each point lies on a side of a rectangle but not in its corner. Also, there are at least two points on every side of a rectangle.

Your task is to choose three points in such a way that:

- exactly two of them belong to the same side of a rectangle;
- the area of a triangle formed by them is maximum possible.

Print the doubled area of this triangle. It can be shown that the doubled area of any triangle formed by lattice points is always an integer.

Input

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

The first line of each testcase contains two integers w and h ($3 \le w, h \le 10^6$) — the coordinates of the corner of a rectangle.

The next two lines contain the description of the points on two horizontal sides. First, an integer k ($2 \le k \le 2 \cdot 10^5$) — the number of points. Then, k integers $x_1 < x_2 < \dots < x_k$ ($0 < x_i < w$) — the x coordinates of the points in the ascending order. The y coordinate for the first line is 0 and for the second line is h.

The next two lines contain the description of the points on two vertical sides. First, an integer k ($2 \le k \le 2 \cdot 10^5$) — the number of points. Then, k integers $y_1 < y_2 < \dots < y_k$ ($0 < y_i < h$) — the y coordinates of the points in the ascending order. The x coordinate for the first line is 0 and for the second line is w.

The total number of points on all sides in all testcases doesn't exceed $2 \cdot 10^5$.

Output

For each testcase print a single integer — the doubled maximum area of a triangle formed by such three points that exactly two of them belong to the same side.

Example

input

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

output

```
25
42
35
```

Note

The points in the first testcase of the example:

- $(1, 0)$, $(2, 0)$;
- $(2, 8)$, $(3, 8)$, $(4, 8)$;
- $(0, 1)$, $(0, 4)$, $(0, 6)$;
- $(5, 4)$, $(5, 5)$.

The largest triangle is formed by points $(0, 1)$, $(0, 6)$ and $(5, 4)$ — its area is $\frac{25}{2}$. Thus, the doubled area is 25. Two points that are on the same side are: $(0, 1)$ and $(0, 6)$.

C. BA-String

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

You are given an integer k and a string s that consists only of characters 'a' (a lowercase Latin letter) and '*' (an asterisk).

Each asterisk should be replaced with several (from 0 to k inclusive) lowercase Latin letters 'b'. Different asterisk can be replaced with different counts of letter 'b'.

The result of the replacement is called a *BA-string*.

Two strings a and b are different if they either have different lengths or there exists such a position i that $a_i \neq b_i$.

A string a is lexicographically smaller than a string b if and only if one of the following holds:

- a is a prefix of b, but $a \ne b$;
- in the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

Now consider all different BA-strings and find the x-th lexicographically smallest of them.

Input

The first line contains a single integer t ($1 \le t \le 2000$) — the number of testcases.

The first line of each testcase contains three integers n, k and x ($1 \le n \le 2000$; $0 \le k \le 2000$; $1 \le x \le 10^{18}$). n is the length of string s.

The second line of each testcase is a string s. It consists of n characters, each of them is either 'a' (a lowercase Latin letter) or '*' (an asterisk).

The sum of n over all testcases doesn't exceed 2000. **For each testcase x doesn't exceed the total number of different BA-strings.** String s contains at least one character 'a'.

Output

For each testcase, print a single string, consisting only of characters 'b' and 'a' (lowercase Latin letters) — the x-th lexicographically smallest BA-string.

Example

input

```
3
2 4 3
a*
4 1 3
a**a
6 3 20
**a***
```

output

```
abb
abba
babbbbbbbbb
```

Note

In the first testcase of the example, BA-strings ordered lexicographically are:

1. a
2. ab
3. abb
4. abbb
5. abbbb

In the second testcase of the example, BA-strings ordered lexicographically are:

1. aa
2. aba
3. abba

Note that string "aba" is only counted once, even though there are two ways to replace asterisks with characters 'b' to get it.

D. Exact Change

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

One day, early in the morning, you decided to buy yourself a bag of chips in the nearby store. The store has chips of n different flavors. A bag of the i-th flavor costs a_i burles.

The store may run out of some flavors, so you'll decide which one to buy after arriving there. But there are two major flaws in this plan:

1. you have only coins of 1, 2 and 3 burles;
2. since it's morning, the store will ask you to pay in exact change, i. e. if you choose the i-th flavor, you'll have to pay *exactly* a_i burles.

Coins are heavy, so you'd like to take the least possible number of coins in total. That's why you are wondering: what is the minimum total number of coins you should take with you, so you can buy a bag of chips of any flavor in exact change?

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 the single integer n ($1 \le n \le 100$) — the number of flavors in the store.

The second line of each test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$) — the cost of one bag of each flavor.

Output

For each test case, print one integer — the minimum number of coins you need to buy one bag of any flavor you'll choose in exact change.

Example

input

```
4
1
1337
3
10 8 10
5
1 2 3 4 5
3
7 77 777
```

output

```
446
4
3
260
```

Note

In the first test case, you should, for example, take with you 445 coins of value 3 and 1 coin of value 2. So, $1337 = 445 \cdot 3 + 1 \cdot 2$.

In the second test case, you should, for example, take 2 coins of value 3 and 2 coins of value 2. So you can pay either exactly $8 = 2 \cdot 3 + 1 \cdot 2$ or $10 = 2 \cdot 3 + 2 \cdot 2$.

In the third test case, it's enough to take 1 coin of value 3 and 2 coins of value 1.

E. Replace the Numbers

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

You have an array of integers (initially empty).

You have to perform q queries. Each query is of one of two types:

- "$1\ x$" — add the element x to the end of the array;
- "$2\ x\ y$" — replace all occurrences of x in the array with y.

Find the resulting array after performing all the queries.

Input

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

Next q lines contain queries (one per line). Each query is of one of two types:

- "$1\ x$" ($1 \le x \le 5 \cdot 10^5$);
- "$2\ x\ y$" ($1 \le x, y \le 5 \cdot 10^5$).

It's guaranteed that there is at least one query of the first type.

Output

In a single line, print k integers — the resulting array after performing all the queries, where k is the number of queries of the first type.

Examples

input

```
7
1 3
1 1
2 1 2
1 2
1 1
1 2
2 1 3
```

output

```
3 2 2 3 2
```

input

```
4
1 1
1 2
1 1
2 2 2
```

output

```
1 2 1
```

input

```
8
2 1 4
1 1
1 4
1 2
2 2 4
2 4 3
1 2
2 2 7
```

output

```
1 3 3 7
```

Note

In the first example, the array changes as follows:

$[] \rightarrow [3] \rightarrow [3, 1] \rightarrow [3, 2] \rightarrow [3, 2, 2] \rightarrow [3, 2, 2, 1] \rightarrow [3, 2, 2, 1, 2] \rightarrow [3, 2, 2, 3, 2]$.

In the second example, the array changes as follows:

$[] \rightarrow [1] \rightarrow [1, 2] \rightarrow [1, 2, 1] \rightarrow [1, 2, 1]$.

In the third example, the array changes as follows:

$[] \rightarrow [] \rightarrow [1] \rightarrow [1, 4] \rightarrow [1, 4, 2] \rightarrow [1, 4, 4] \rightarrow [1, 3, 3] \rightarrow [1, 3, 3, 2] \rightarrow [1, 3, 3, 7]$.

F. Bipartite Array

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

You are given a permutation p consisting of n integers $1, 2, \dots, n$ (a permutation is an array where each element from 1 to n occurs exactly once).

Let's call an array a *bipartite* if the following undirected graph is bipartite:

- the graph consists of n vertices;
- two vertices i and j are connected by an edge if $i < j$ and $a_i > a_j$.

Your task is to find a *bipartite* array of integers a of size n, such that $a_i = p_i$ or $a_i = -p_i$, or report that no such array exists. If there are multiple answers, print any of them.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 10^6$) — the size of the permutation.

The second line contains n integers $p_1, p_2, \dots, p_n$.

The sum of n over all test cases doesn't exceed 10^6.

Output

For each test case, print the answer in the following format. If such an array a does not exist, print "NO" in a single line. Otherwise, print "YES" in the first line and n integers — array a in the second line.

Example

input

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

output

```
YES
1 2 3
NO
YES
-4 -1 -3 -2
YES
-3 -2 1 6 7 -8 -5 -4
```

G. Subsequences Galore

time limit per test: 10 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

For a sequence of strings $[t_1, t_2, \dots, t_m]$, let's define the function $f([t_1, t_2, \dots, t_m])$ as the number of different strings (**including the empty string**) that are subsequences of **at least one** string t_i. $f([]) = 0$ (i. e. the number of such strings for an empty sequence is 0).

You are given a sequence of strings $[s_1, s_2, \dots, s_n]$. Every string in this sequence consists of lowercase Latin letters and is **sorted** (i. e., each string begins with several (maybe zero) characters a, then several (maybe zero) characters b, ..., ends with several (maybe zero) characters z).

For each of 2^n subsequences of $[s_1, s_2, \dots, s_n]$, calculate the value of the function f modulo 998244353.

Input

The first line contains one integer n ($1 \le n \le 23$) — the number of strings.

Then n lines follow. The i-th line contains the string s_i ($1 \le |s_i| \le 2 \cdot 10^4$), consisting of lowercase Latin letters. Each string s_i is sorted.

Output

Since printing up to 2^{23} integers would be really slow, you should do the following:

For each of the 2^n subsequences (which we denote as $[s_{i_1}, s_{i_2}, \dots, s_{i_k}]$), calculate $f([s_{i_1}, s_{i_2}, \dots, s_{i_k}])$, take it modulo 998244353, then multiply it by $k \cdot (i_1 + i_2 + \dots + i_k)$. Print the XOR of all 2^n integers you get.

The indices $i_1, i_2, \dots, i_k$ in the description of each subsequences are 1-indexed (i. e. are from 1 to n).

Examples

input
```
3
a
b
c
```
output
```
92
```

input
```
2
aa
a
```
output
```
21
```

input
```
2
a
a
```
output
```
10
```

input
```
2
abcd
aabb
```
output
```
124
```

input
```
3
ddd
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaabbbbbbbbbbbcccccccccccciiiiiiiiiiiiiiiiiiiiiioooooooooooqqqqqqqqqqqqqqqqqqvvvvvzzzzzzzzzzzz
```
output
```
15706243380
```