Codeforces Round #723 (Div. 2)

A. Mean Inequality

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array a of $2n$ **distinct** integers. You want to arrange the elements of the array in a circle such that no element is equal to the the arithmetic mean of its 2 neighbours.

More formally, find an array b, such that:

- b is a permutation of a.
- For every i from 1 to $2n$, $b_i \neq \frac{b_{i-1}+b_{i+1}}{2}$, where $b_0 = b_{2n}$ and $b_{2n+1} = b_1$.

It can be proved that under the constraints of this problem, such array b always exists.

Input

The first line of input contains a single integer t $(1 \leq t \leq 1000)$ — the number of testcases. The description of testcases follows.

The first line of each testcase contains a single integer n $(1 \leq n \leq 25)$.

The second line of each testcase contains $2n$ integers $a_1, a_2, \ldots, a_{2n}$ $(1 \leq a_i \leq 10^9)$ — elements of the array.

Note that there is no limit to the sum of n over all testcases.

Output

For each testcase, you should output $2n$ integers, $b_1, b_2, \ldots b_{2n}$, for which the conditions from the statement are satisfied.

Example

input

```
3
3
1 2 3 4 5 6
2
123 456 789 10
1
6 9
```

output

```
3 1 4 2 5 6
123 10 456 789
9 6
```

Note

In the first testcase, array $[3, 1, 4, 2, 5, 6]$ works, as it's a permutation of $[1, 2, 3, 4, 5, 6]$, and $\frac{3+4}{2} \neq 1$, $\frac{1+2}{2} \neq 4$, $\frac{4+5}{2} \neq 2$, $\frac{2+6}{2} \neq 5$, $\frac{5+3}{2} \neq 6$, $\frac{6+1}{2} \neq 3$.

B. I Hate 1111

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an integer x. Can you make x by summing up some number of $11, 111, 1111, 11111, \ldots$? (You can use any number among them any number of times).

For instance,

- $33 = 11 + 11 + 11$
- $144 = 111 + 11 + 11 + 11$

Input

The first line of input contains a single integer t $(1 \leq t \leq 10000)$ — the number of testcases.

The first and only line of each testcase contains a single integer x $(1 \leq x \leq 10^9)$ — the number you have to make.

Output

For each testcase, you should output a single string. If you can make x, output "YES" (without quotes). Otherwise, output "NO".

You can print each letter of "YES" and "NO" in any case (upper or lower).

Example

input

```
3
33
144
69
```

output

```
YES
YES
NO
```

Note

Ways to make 33 and 144 were presented in the statement. It can be proved that we can't present 69 this way.

C1. Potions (Easy Version)

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is the easy version of the problem. The only difference is that in this version $n \leq 2000$. You can make hacks only if both versions of the problem are solved.

There are n potions in a line, with potion 1 on the far left and potion n on the far right. Each potion will increase your health by a_i when drunk. a_i can be negative, meaning that potion will decrease will health.

You start with 0 health and you will walk from left to right, from first potion to the last one. At each potion, you may choose to drink it or ignore it. **You must ensure that your health is always non-negative**.

What is the largest number of potions you can drink?

Input

The first line contains a single integer n ($1 \leq n \leq 2000$) — the number of potions.

The next line contains n integers a_1, a_2, ... ,a_n ($-10^9 \leq a_i \leq 10^9$) which represent the change in health after drinking that potion.

Output

Output a single integer, the maximum number of potions you can drink without your health becoming negative.

Example

input

```
6
4 -4 1 -3 1 -3
```

output

```
5
```

Note

For the sample, you can drink 5 potions by taking potions 1, 3, 4, 5 and 6. It is not possible to drink all 6 potions because your health will go negative at some point

C2. Potions (Hard Version)

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is the hard version of the problem. The only difference is that in this version $n \leq 200000$. You can make hacks only if both versions of the problem are solved.

There are n potions in a line, with potion 1 on the far left and potion n on the far right. Each potion will increase your health by a_i when drunk. a_i can be negative, meaning that potion will decrease will health.

You start with 0 health and you will walk from left to right, from first potion to the last one. At each potion, you may choose to drink it or ignore it. **You must ensure that your health is always non-negative**.

What is the largest number of potions you can drink?

Input

The first line contains a single integer n $(1 \le n \le 200000)$ — the number of potions.

The next line contains n integers a_1, a_2, ... ,a_n $(-10^9 \le a_i \le 10^9)$ which represent the change in health after drinking that potion.

Output

Output a single integer, the maximum number of potions you can drink without your health becoming negative.

Example

input

```
6
4 -4 1 -3 1 -3
```

output

```
5
```

Note

For the sample, you can drink 5 potions by taking potions 1, 3, 4, 5 and 6. It is not possible to drink all 6 potions because your health will go negative at some point

D. Kill Anton

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

After rejecting 10^{100} data structure problems, Errorgorn is very angry at Anton and decided to kill him.

Anton's DNA can be represented as a string a which only contains the characters "ANTON" (there are only 4 distinct characters).

Errorgorn can change Anton's DNA into string b which must be a **permutation** of a. However, Anton's body can defend against this attack. In 1 second, his body can swap 2 **adjacent** characters of his DNA to transform it back to a. Anton's body is smart and will use the minimum number of moves.

To maximize the chance of Anton dying, Errorgorn wants to change Anton's DNA the string that maximizes the time for Anton's body to revert his DNA. But since Errorgorn is busy making more data structure problems, he needs your help to find the best string B. Can you help him?

Input

The first line of input contains a single integer t $(1 \le t \le 100000)$ — the number of testcases.

The first and only line of each testcase contains 1 string a $(1 \le |a| \le 100000)$. a consists of only the characters "A", "N", "O" and "T".

It is guaranteed that the sum of $|a|$ over all testcases does not exceed 100000.

Output

For each testcase, print a single string, b. If there are multiple answers, you can output any one of them. b must be a permutation of the string a.

Example

input

```
4
ANTON
NAAN
AAAAAA
OAANTTON
```

output

```
NNOTA
AANN
AAAAAA
TNNTAOOA
```

Note

For the first testcase, it takes 7 seconds for Anton's body to transform NNOTA to ANTON:

NNOTA $\to$ NNOAT $\to$ NNAOT $\to$ NANOT $\to$ NANTO $\to$ ANNTO $\to$ ANTNO $\to$ ANTON.

Note that you cannot output strings such as AANTON, ANTONTRYGUB, AAAAA and anton as it is not a permutation of ANTON.

For the second testcase, it takes 2 seconds for Anton's body to transform AANN to NAAN. Note that other strings such as NNAA and ANNA will also be accepted.

E. Oolimry and Suffix Array

time limit per test: 1 second

memory limit per test: 256 megabytes
input: standard input
output: standard output

Once upon a time, Oolimry saw a suffix array. He wondered how many strings can produce this suffix array.

More formally, given a suffix array of length n and having an alphabet size k, count the number of strings that produce such a suffix array.

Let s be a string of length n. Then the i-th suffix of s is the substring $s[i \dots n-1]$. A suffix array is the array of integers that represent the starting indexes of all the suffixes of a given string, after the suffixes are sorted in the lexicographic order. For example, the suffix array of `oolimry` is $[3, 2, 4, 1, 0, 5, 6]$ as the array of sorted suffixes is $[\texttt{imry}, \texttt{limry}, \texttt{mry}, \texttt{olimry}, \texttt{oolimry}, \texttt{ry}, \texttt{y}]$.

A string x is lexicographically smaller than string y, if either x is a prefix of y (and $x \neq y$), or there exists such i that $x_i < y_i$, and for any $1 \leq j < i$, $x_j = y_j$.

Input

The first line contain 2 integers n and k ($1 \leq n \leq 200000, 1 \leq k \leq 200000$) — the length of the suffix array and the alphabet size respectively.

The second line contains n integers $s_0, s_1, s_2, \dots, s_{n-1}$ ($0 \leq s_i \leq n-1$) **where s_i is the i-th element of the suffix array** i.e. the starting position of the i-th lexicographically smallest suffix. It is guaranteed that for all $0 \leq i < j \leq n-1$, $s_i \neq s_j$.

Output

Print how many strings produce such a suffix array. Since the number can be very large, print the answer modulo 998244353.

Examples

input
```
3 2
0 2 1
```
output
```
1
```

input
```
5 1
0 1 2 3 4
```
output
```
0
```

input
```
6 200000
0 1 2 3 4 5
```
output
```
822243495
```

input
```
7 6
3 2 4 1 0 5 6
```
output
```
36
```

Note

In the first test case, "abb" is the only possible solution.

In the second test case, it can be easily shown no possible strings exist as all the letters have to be equal.

In the fourth test case, one possible string is "ddbacef".

Please remember to print your answers modulo 998244353.

F. Median Queries

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive problem.

There is a secret permutation p (1-indexed) of numbers from 1 to n. More formally, for $1 \leq i \leq n$, $1 \leq p[i] \leq n$ and for $1 \leq i < j \leq n$, $p[i] \neq p[j]$. **It is known that $p[1] < p[2]$.**

In 1 query, you give 3 **distinct** integers a, b, c ($1 \le a, b, c \le n$), and receive the **median** of $\{|p[a] - p[b]|, |p[b] - p[c]|, |p[a] - p[c]|\}$.

In this case, the median is the 2-nd element (1-indexed) of the sequence when sorted in non-decreasing order. The median of $\{4, 6, 2\}$ is 4 and the median of $\{0, 123, 33\}$ is 33.

Can you find the secret permutation in not more than $2n + 420$ queries?

Note: the grader is not adaptive: the permutation is fixed before any queries are made.

Input

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

The first line of each testcase consists of a single integer n $(20 \le n \le 100000)$ — the length of the secret permutation.

It is guaranteed that the sum of n over all test cases does not exceed 100000.

Interaction

For each testcase, you begin the interaction by reading n.

To perform a query, output "`? a b c`" where a, b, c is the 3 indices you want to use for the query.

Numbers have to satisfy $1 \le a, b, c \le n$ and $a \ne b$,$b \ne c$,$a \ne c$.

For each query, you will receive a single integer x: the **median** of $\{|p[a] - p[b]|, |p[b] - p[c]|, |p[a] - p[c]|\}$.

In case your query is invalid or you asked more than $2n + 420$ queries, the interactor will print "−1" and will finish interaction. You will receive **Wrong answer** verdict. Make sure to exit immediately to avoid getting other verdicts.

When you have determined the secret permutation, output "`! p[1] p[2] ... p[n]`". If the secret permutation is correct, the interactor will print "`1`". Otherwise, the interactor will print "`-1`" and will finish interaction. You will receive **Wrong answer** verdict. Make sure to exit immediately to avoid getting other verdicts.

After printing a query do not forget to output the end of line and flush the output. Otherwise, you will get `Idleness limit exceeded` verdict.

To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

Hacks:

To hack, use the following format of test:

The first line should contain a single integer t ($1 \le t \le 1000$) — the number of testcases.

The first line of each testcase should contain a single integer n ($20 \le n \le 100000$) — the length of the secret permutation.

The following line of should contain n integers $p[1], p[2], p[3], \ldots, p[n]$. $p[1] < p[2]$ and p must be a permutation of integers from 1 to n.

You must ensure that the sum of n over all testcases does not exceed 100000.

Example

input

```
1
20

6

9

1
```

output

```


? 1 5 2

? 20 19 2

! 9 10 19 7 16 18 11 14 15 6 20 8 17 4 5 3 12 2 13 1
```

Note

The secret permutation is $\{9, 10, 19, 7, 16, 18, 11, 14, 15, 6, 20, 8, 17, 4, 5, 3, 12, 2, 13, 1\}$.

For the first query, the values of (a, b, c) is $(1, 5, 2)$. Since $p[1] = 9$, $p[5] = 16$ and $p[2] = 10$. The return value is the median of $\{|9 - 16|, |16 - 10|, |9 - 10|\}$ which is 6.

For the second query, the values of (a, b, c) is $(20, 19, 2)$. Since $p[20] = 1$, $p[19] = 13$ and $p[2] = 10$. The return value is the median of $\{|1 - 13|, |13 - 10|, |1 - 10|\}$ which is 9.

By some miracle, we have figured out that the secret permutation is $\{9, 10, 19, 7, 16, 18, 11, 14, 15, 6, 20, 8, 17, 4, 5, 3, 12, 2, 13, 1\}$. We output it and receive 1 from the interactor, meaning that we have guessed the secret permutation correctly.