Codeforces Round #785 (Div. 2)

A. Subtle Substring Subtraction

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

Alice and Bob are playing a game with strings. There will be t rounds in the game. In each round, there will be a string s consisting of lowercase English letters.

Alice moves first and both the players take alternate turns. **Alice is allowed to remove any substring of even length (possibly empty) and Bob is allowed to remove any substring of odd length from s.**

More formally, if there was a string $s = s_1 s_2 \ldots s_k$ the player can choose a substring $s_l s_{l+1} \ldots s_{r-1} s_r$ with length of corresponding parity and remove it. After that the string will become $s = s_1 \ldots s_{l-1} s_{r+1} \ldots s_k$.

After the string becomes empty, the round ends and each player calculates his/her score for this round. The score of a player is the sum of values of all characters removed by him/her. The value of `a` is 1, the value of `b` is 2, the value of `c` is 3, ..., and the value of `z` is 26. The player with higher score wins the round. For each round, determine the winner and the difference between winner's and loser's scores. Assume that both players play optimally to maximize their score. It can be proved that a draw is impossible.

Input

The first line of input contains a single integer t ($1 \le t \le 5 \cdot 10^4$) denoting the number of rounds.

Each of the next t lines contain a single string s ($1 \le |s| \le 2 \cdot 10^5$) consisting of lowercase English letters, denoting the string used for the round. Here $|s|$ denotes the length of the string s.

It is guaranteed that the sum of $|s|$ over all rounds does not exceed $2 \cdot 10^5$.

Output

For each round, print a single line containing a string and an integer. If Alice wins the round, the string must be "Alice". If Bob wins the round, the string must be "Bob". The integer must be the difference between their scores assuming both players play optimally.

Example

input
5 aba abc cba n codeforces
output
Alice 2 Alice 4 Alice 4 Bob 14 Alice 93

Note

For the first round, "aba" $\xrightarrow{\text{Alice}}$ "aba" $\rightarrow$ "a" $\xrightarrow{\text{Bob}}$ "a" $\rightarrow$ "". Alice's total score is $1 + 2 = 3$. Bob's total score is 1.

For the second round, "abc" $\xrightarrow{\text{Alice}}$ "abc" $\rightarrow$ "a" $\xrightarrow{\text{Bob}}$ "a" $\rightarrow$ "". Alice's total score is $2 + 3 = 5$. Bob's total score is 1.

For the third round, "cba" $\xrightarrow{\text{Alice}}$ "cba" $\rightarrow$ "a" $\xrightarrow{\text{Bob}}$ "a" $\rightarrow$ "". Alice's total score is $3 + 2 = 5$. Bob's total score is 1.

For the fourth round, "n" $\xrightarrow{\text{Alice}}$ "n" $\rightarrow$ "n" $\xrightarrow{\text{Bob}}$ "n" $\rightarrow$ "". Alice's total score is 0. Bob's total score is 14.

For the fifth round, "codeforces" $\xrightarrow{\text{Alice}}$ "codeforces" $\rightarrow$ "". Alice's total score is $3 + 15 + 4 + 5 + 6 + 15 + 18 + 3 + 5 + 19 = 93$. Bob's total score is 0.

B. A Perfectly Balanced String?

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

Let's call a string s perfectly balanced if for all possible triplets (t, u, v) such that t is a non-empty substring of s and u and v are characters present in s, the difference between the frequencies of u and v in t is not more than 1.

For example, the strings "aba" and "abc" are perfectly balanced but "abb" is not because for the triplet ("bb",'a','b'), the condition is not satisfied.

You are given a string s consisting of lowercase English letters only. Your task is to determine whether s is perfectly balanced or not.

A string b is called a substring of another string a if b can be obtained by deleting some characters (possibly 0) from the start and some characters (possibly 0) from the end of a.

Input

The first line of input contains a single integer t ($1 \leq t \leq 2 \cdot 10^4$) denoting the number of testcases.

Each of the next t lines contain a single string s ($1 \leq |s| \leq 2 \cdot 10^5$), consisting of lowercase English letters.

It is guaranteed that the sum of $|s|$ over all testcases does not exceed $2 \cdot 10^5$.

Output

For each test case, print "YES" if s is a perfectly balanced string, and "NO" otherwise.

You may print each letter in any case (for example, "YES", "Yes", "yes", "yEs" will all be recognized as positive answer).

Example

input
5 aba abb abc aaaaa abcba
output
YES NO YES YES NO

Note

Let $f_t(c)$ represent the frequency of character c in string t.

For the first testcase we have

t	$f_t(a)$	$f_t(b)$
a	1	0
ab	1	1
aba	2	1
b	0	1
ba	1	1

It can be seen that for any substring t of s, the difference between $f_t(a)$ and $f_t(b)$ is not more than 1. Hence the string s is perfectly balanced.

For the second testcase we have

t	$f_t(a)$	$f_t(b)$
a	1	0
ab	1	1
abb	1	2
b	0	1
bb	0	2

It can be seen that for the substring $t = bb$, the difference between $f_t(a)$ and $f_t(b)$ is 2 which is greater than 1. Hence the string s is not perfectly balanced.

For the third testcase we have

t	$f_t(a)$	$f_t(b)$	$f_t(c)$
a	1	0	0
ab	1	1	0
abc	1	1	1
b	0	1	0
bc	0	1	1
c	0	0	1

It can be seen that for any substring t of s and any two characters $u, v \in \{a, b, c\}$, the difference between $f_t(u)$ and $f_t(v)$ is not more than 1. Hence the string s is perfectly balanced.

C. Palindrome Basis

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

You are given a positive integer n. Let's call some positive integer a without leading zeroes palindromic if it remains the same after reversing the order of its digits. Find the number of distinct ways to express n as a sum of positive palindromic integers. Two ways are considered different if the frequency of at least one palindromic integer is different in them. For example, $5 = 4 + 1$ and $5 = 3 + 1 + 1$ are considered different but $5 = 3 + 1 + 1$ and $5 = 1 + 3 + 1$ are considered the same.

Formally, you need to find the number of distinct multisets of positive palindromic integers the sum of which is equal to n.

Since the answer can be quite large, print it modulo $10^9 + 7$.

Input

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

Each testcase contains a single line of input containing a single integer n ($1 \le n \le 4 \cdot 10^4$) — the required sum of palindromic integers.

Output

For each testcase, print a single integer denoting the required answer modulo $10^9 + 7$.

Example

input
2 5 12
output
7 74

Note

For the first testcase, there are 7 ways to partition 5 as a sum of positive palindromic integers:

- $5 = 1 + 1 + 1 + 1 + 1$
- $5 = 1 + 1 + 1 + 2$
- $5 = 1 + 2 + 2$
- $5 = 1 + 1 + 3$
- $5 = 2 + 3$
- $5 = 1 + 4$
- $5 = 5$

For the second testcase, there are total 77 ways to partition 12 as a sum of positive integers but among them, the partitions $12 = 2 + 10$, $12 = 1 + 1 + 10$ and $12 = 12$ are not valid partitions of 12 as a sum of positive palindromic integers because 10 and 12 are not palindromic. So, there are 74 ways to partition 12 as a sum of positive palindromic integers.

D. Lost Arithmetic Progression

time limit per test: 1 second

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

Long ago, you thought of two finite arithmetic progressions A and B. Then you found out another sequence C containing all elements common to both A and B. It is not hard to see that C is also a finite arithmetic progression. After many years, you forgot what A was but remember B and C. You are, for some reason, determined to find this lost arithmetic progression. Before you begin this eternal search, you want to know how many different finite arithmetic progressions exist which can be your lost progression A.

Two arithmetic progressions are considered different if they differ in their first term, common difference or number of terms.

It may be possible that there are infinitely many such progressions, in which case you won't even try to look for them! Print -1 in all such cases.

Even if there are finite number of them, the answer might be very large. So, you are only interested to find the answer modulo $10^9 + 7$.

Input

The first line of input contains a single integer t ($1 \le t \le 100$) denoting the number of testcases.

The first line of each testcase contains three integers b, q and y ($-10^9 \le b \le 10^9$, $1 \le q \le 10^9$, $2 \le y \le 10^9$) denoting the first term, common difference and number of terms of B respectively.

The second line of each testcase contains three integers c, r and z ($-10^9 \le c \le 10^9$, $1 \le r \le 10^9$, $2 \le z \le 10^9$) denoting the first term, common difference and number of terms of C respectively.

Output

For each testcase, print a single line containing a single integer.

If there are infinitely many finite arithmetic progressions which could be your lost progression A, print -1.

Otherwise, print the number of finite arithmetic progressions which could be your lost progression A modulo $10^9 + 7$. In particular, if there are no such finite arithmetic progressions, print 0.

Example

input
8 -3 1 7 -1 2 4 -9 3 11 0 6 3 2 5 5 7 5 4 2 2 11 10 5 3 0 2 9 2 4 3 -11 4 12 1 12 2 -27 4 7 -17 8 2 -8400 420 1000000000 0 4620 10
output
0 10 -1 0 -1 21 0 273000

Note

For the first testcase, $B = \{-3, -2, -1, 0, 1, 2, 3\}$ and $C = \{-1, 1, 3, 5\}$. There is no such arithmetic progression which can be equal to A because 5 is not present in B and for any A, 5 should not be present in C also.

For the second testcase, $B = \{-9, -6, -3, 0, 3, 6, 9, 12, 15, 18, 21\}$ and $C = \{0, 6, 12\}$. There are 10 possible arithmetic progressions which can be A:

- $\{0, 6, 12\}$
- $\{0, 2, 4, 6, 8, 10, 12\}$
- $\{0, 2, 4, 6, 8, 10, 12, 14\}$
- $\{0, 2, 4, 6, 8, 10, 12, 14, 16\}$
- $\{-2, 0, 2, 4, 6, 8, 10, 12\}$
- $\{-2, 0, 2, 4, 6, 8, 10, 12, 14\}$
- $\{-2, 0, 2, 4, 6, 8, 10, 12, 14, 16\}$
- $\{-4, -2, 0, 2, 4, 6, 8, 10, 12\}$
- $\{-4, -2, 0, 2, 4, 6, 8, 10, 12, 14\}$

- $\{-4, -2, 0, 2, 4, 6, 8, 10, 12, 14, 16\}$

For the third testcase, $B = \{2, 7, 12, 17, 22\}$ and $C = \{7, 12, 17, 22\}$. There are infinitely many arithmetic progressions which can be A like:

- $\{7, 12, 17, 22\}$
- $\{7, 12, 17, 22, 27\}$
- $\{7, 12, 17, 22, 27, 32\}$
- $\{7, 12, 17, 22, 27, 32, 37\}$
- $\{7, 12, 17, 22, 27, 32, 37, 42\}$
- $\ldots$

E. Power or XOR?

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

The symbol $\wedge$ is quite ambiguous, especially when used without context. Sometimes it is used to denote a `power` ($a \wedge b = a^b$) and sometimes it is used to denote the XOR operation ($a \wedge b = a \oplus b$).

You have an ambiguous expression $E = A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_n$. You can replace each $\wedge$ symbol with either a `Power` operation or a `XOR` operation to get an unambiguous expression E'.

The value of this expression E' is determined according to the following rules:

- All `Power` operations are performed before any `XOR` operation. In other words, the `Power` operation takes precedence over `XOR` operation. For example, 4 `XOR` 6 `Power` $2 = 4 \oplus (6^2) = 4 \oplus 36 = 32$.
- Consecutive powers are calculated from **left to right**. For example, 2 `Power` 3 `Power` $4 = (2^3)^4 = 8^4 = 4096$.

You are given an array B of length n and an integer k. The array A is given by $A_i = 2^{B_i}$ and the expression E is given by $E = A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_n$. You need to find the XOR of the values of all possible unambiguous expressions E' which can be obtained from E and has at least k $\wedge$ symbols used as `XOR` operation. Since the answer can be very large, you need to find it modulo $2^{2^{20}}$. Since this number can also be very large, you need to print its binary representation **without leading zeroes**. If the answer is equal to 0, print 0.

Input

The first line of input contains two integers n and k $(1 \leq n \leq 2^{20}, 0 \leq k < n)$.

The second line of input contains n integers $B_1, B_2, \ldots, B_n$ $(1 \leq B_i < 2^{20})$.

Output

Print a single line containing a binary string without leading zeroes denoting the answer to the problem. If the answer is equal to 0, print 0.

Examples

input
3 2 3 1 2
output
1110

input
3 1 3 1 2
output
1010010

input
3 0 3 1 2
output
1000000000000000001010010

input
2 1 1 1
output
0

Note

For each of the testcases 1 to 3, $A = \{2^3, 2^1, 2^2\} = \{8, 2, 4\}$ and $E = 8 \wedge 2 \wedge 4$.

For the first testcase, there is only one possible valid unambiguous expression $E' = 8 \oplus 2 \oplus 4 = 14 = (1110)_2$.

For the second testcase, there are three possible valid unambiguous expressions E':

- $8 \oplus 2 \oplus 4 = 14$
- $8^2 \oplus 4 = 64 \oplus 4 = 68$
- $8 \oplus 2^4 = 8 \oplus 16 = 24$

XOR of the values of all of these is $14 \oplus 68 \oplus 24 = 82 = (1010010)_2$.
For the third testcase, there are four possible valid unambiguous expressions E':

- $8 \oplus 2 \oplus 4 = 14$
- $8^2 \oplus 4 = 64 \oplus 4 = 68$
- $8 \oplus 2^4 = 8 \oplus 16 = 24$
- $(8^2)^4 = 64^4 = 2^{24} = 16777216$

XOR of the values of all of these is $14 \oplus 68 \oplus 24 \oplus 16777216 = 16777298 = (1000000000000000001010010)_2$.
For the fourth testcase, $A = \{2, 2\}$ and $E = 2 \wedge 2$. The only possible valid unambiguous expression $E' = 2 \oplus 2 = 0 = (0)_2$.

F. Anti-Theft Road Planning

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

This is an interactive problem.

A city has n^2 buildings divided into a grid of n rows and n columns. You need to build a road of some length $D(A, B)$ of your choice between each pair of adjacent by side buildings A and B. Due to budget limitations and legal restrictions, the length of each road must be a positive integer and **the total length of all roads should not exceed** $48\,000$.

There is a thief in the city who will start from the topmost, leftmost building (in the first row and the first column) and roam around the city, occasionally stealing artifacts from some of the buildings. He can move from one building to another adjacent building by travelling through the road which connects them.

You are unable to track down what buildings he visits and what path he follows to reach them. But there is one tracking mechanism in the city. The tracker is capable of storing a single integer x which is initially 0. Each time the thief travels from a building A to another adjacent building B through a road of length $D(A, B)$, the tracker changes x to $x \oplus D(A, B)$. Each time the thief steals from a building, the tracker reports the value x stored in it and resets it back to 0.

It is known beforehand that the thief will steal in exactly k buildings but you will know the values returned by the tracker only after the thefts actually happen. Your task is to choose the lengths of roads in such a way that no matter what strategy or routes the thief follows, you will be able to exactly tell the location of all the buildings where the thefts occurred from the values returned by the tracker.

Interaction

First read a single line containing two integers n $(2 \le n \le 32)$ and k $(1 \le k \le 1024)$ denoting the number of rows and number of thefts respectively.

Let's denote the j-th building in the i-th row by $B_{i,j}$.

Then print n lines each containing $n - 1$ integers. The j-th integer of the i-th line must be the value of $D(B_{i,j}, B_{i,j+1})$.

Then print $n - 1$ lines each containing n integers. The j-th integer of the i-th line must be the value of $D(B_{i,j}, B_{i+1,j})$.

Remember that the total length of the roads must not exceed $48\,000$.

Then answer k queries. First read the value x returned by the tracker. Then print two integers denoting the row number and column number of the building where the theft occurred. After that you will be able to answer the next query (if such exists).

After printing the answers do not forget to output end of line and flush the output buffer. Otherwise you will get the verdict `Idleness limit exceeded`. To flush the buffer, use:

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

Hacks

You cannot make hacks in this problem.

Example

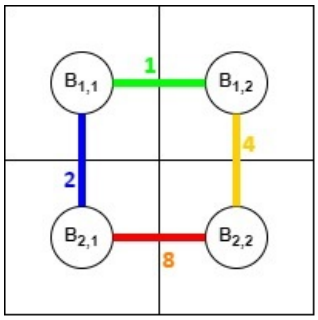
input
2 4   14 1 14 3

output
1 8 2 4 1 2 1 1 1 2 2 1

Note

For the sample test, $n = 2$ and $k = 4$.

You choose to build the roads of the following lengths:



The thief follows the following strategy:

1. Start at $B_{1,1}$.
2. Move Right to $B_{1,2}$.
3. Move Down to $B_{2,2}$.
4. Move Left to $B_{2,1}$.
5. Move Up to $B_{1,1}$.
6. Move Right to $B_{1,2}$.
7. Steal from $B_{1,2}$.
8. Move Left to $B_{1,1}$.
9. Steal from $B_{1,1}$.
10. Move Down to $B_{2,1}$.
11. Move Right to $B_{2,2}$.
12. Move Up to $B_{1,2}$.
13. Steal from $B_{1,2}$.
14. Move Left to $B_{1,1}$.
15. Move Down to $B_{2,1}$.
16. Steal from $B_{2,1}$.

The tracker responds in the following way:

1. Initialize $x = 0$.
2. Change x to $x \oplus 1 = 0 \oplus 1 = 1$.
3. Change x to $x \oplus 4 = 1 \oplus 4 = 5$.
4. Change x to $x \oplus 8 = 5 \oplus 8 = 13$.
5. Change x to $x \oplus 2 = 13 \oplus 2 = 15$.
6. Change x to $x \oplus 1 = 15 \oplus 1 = 14$.
7. Return $x = 14$ and re-initialize $x = 0$.
8. Change x to $x \oplus 1 = 0 \oplus 1 = 1$.
9. Return $x = 1$ and re-initialize $x = 0$.

10. Change x to $x \oplus 2 = 0 \oplus 2 = 2$.
11. Change x to $x \oplus 8 = 2 \oplus 8 = 10$.
12. Change x to $x \oplus 4 = 10 \oplus 4 = 14$.
13. Return $x = 14$ and re-initialize $x = 0$.
14. Change x to $x \oplus 1 = 0 \oplus 1 = 1$.
15. Change x to $x \oplus 2 = 1 \oplus 2 = 3$.
16. Return $x = 3$ and re-initialize $x = 0$.
