A. Special Characters

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

You are given an integer n.

Your task is to build a string of uppercase Latin letters. There must be exactly n special characters in this string. Let's call a character *special* if it is equal to exactly one of its neighbors.

For example, there are 6 special characters in the AAABAACC string (at positions: 1, 3, 5, 6, 7 and 8).

Print any suitable string or report that there is no such string.

Input

The first line contains a single integer t ($1 \le t \le 50$) — the number of test cases.

The only line of each test case contains a single integer n ($1 \le n \le 50$).

Output

For each test case, print the answer as follows:

- if there is no suitable string, print one line containing the string NO;
- otherwise, print two lines. The first line should contain the string YES; on the second line print a string of length **at most** 200 — the answer itself (it can be shown that if some answers exist, then there is an answer of length at most 200). If there are several solutions, print any of them.

Standard Input	Standard Output
3 6 1 2	YES AAABAACC NO YES MM

,

B. Array Fix

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

You are given an integer array a of length n.

You can perform the following operation any number of times (possibly zero): take any element of the array a, which is at least 10, delete it, and instead insert the digits that element consisted of in the same position, in order they appear in that element.

For example:

- if we apply this operation to the 3-rd element of the array $[12, 3, 45, 67]$, then the array becomes $[12, 3, 4, 5, 67]$.
- if we apply this operation to the 2-nd element of the array $[2, 10]$, then the array becomes $[2, 1, 0]$.

Your task is to determine whether it is possible to make a sorted in non-descending order using the aforementioned operation **any number of times (possibly zero)**. In other words, you have to determine if it is possible to transform the array a in such a way that $a_1 \le a_2 \le \dots \le a_k$, where k is the current length of the array a.

Input

The first line contains a single integer t ($1 \le t \le 10^3$) — the number of test cases.

Each test case consists of two lines:

- the first line contains a single integer n ($2 \le n \le 50$).
- the second line contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le 99$).

Output

For each test case, print YES if it is possible to make a sorted in non-decreasing order using the aforementioned operation; otherwise, print NO.

You can print each letter in any case. For example, yes, Yes, YeS will all be recognized as a positive answer.

Standard Input	Standard Output
3 4 12 3 45 67 3 12 28 5 2 0 0	YES NO YES

Note

In the first example, you can split the first element, then the array becomes $[1, 2, 3, 45, 67]$.

In the second example, there is no way to get a sorted array.

In the third example, the array is already sorted.

,

C. Arrow Path

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

There is a grid, consisting of 2 rows and n columns. The rows are numbered from 1 to 2 from top to bottom. The columns are numbered from 1 to n from left to right. Each cell of the grid contains an arrow pointing either to the left or to the right. No arrow points outside the grid.

There is a robot that starts in a cell $(1, 1)$. Every second, the following two actions happen one after another:

1. Firstly, the robot moves left, right, down or up (**it can't try to go outside the grid, and can't skip a move**);
2. then it moves along the arrow that is placed in the current cell (the cell it ends up after its move).

Your task is to determine whether the robot can reach the cell $(2, n)$.

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 ($2 \le n \le 2 \cdot 10^5$).

The second line contains a string consisting of exactly n characters < and/or > — the first row of the grid.

The third line contains a string consisting of exactly n characters < and/or > — the second row of the grid.

Additional constraints on the input:

- n is even;
- there are no arrows pointing outside the grid;
- the sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print YES if the robot can reach the cell $(2, n)$; otherwise, print NO.

You can print each letter in any case. For example, yes, Yes, YeS will all be recognized as positive answer.

Standard Input	Standard Output
4 4 >><< >>>< 2 > > 4 >>>< >><< 6 >><<><	YES YES NO YES

><>>><	

Note

In the first example, one of the possible paths looks as follows:
$(1, 1) \to (1, 2) \to (1, 3) \to (2, 3) \to (2, 4)$.

In the second example, one of the possible paths looks as follows: $(1, 1) \to (2, 1) \to (2, 2)$.

In the third example, there is no way to reach the cell $(2, 4)$.

In the fourth example, one of the possible paths looks as follows:
$(1, 1) \to (2, 1) \to (2, 2) \to (1, 2) \to (1, 3) \to (2, 3) \to (2, 4) \to (2, 5) \to (2, 6)$.

,

D. Tandem Repeats?

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

You are given a string s, consisting of lowercase Latin letters and/or question marks.

A *tandem repeat* is a string of an even length such that its first half is equal to its second half.

A string a is a substring of a string b if a can be obtained from b by the deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

Your goal is to replace each question mark with some lowercase Latin letter in such a way that the length of the longest substring that is a tandem repeat is maximum possible.

Input

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

The only line of each testcase contains a string s ($1 \le |s| \le 5000$), consisting only of lowercase Latin letters and/or question marks.

The total length of the strings over all testcases doesn't exceed 5000.

Output

For each testcase, print a single integer — the maximum length of the longest substring that is a tandem repeat after you replace each question mark in the string with some lowercase Latin letter.

If it's impossible to make any tandem repeat substrings in the string, print 0.

Standard Input	Standard Output
4 zaabaabz ????? code?????s codeforces	6 4 10 0

,

E. Clique Partition

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 512 megabytes

You are given two integers, n and k. There is a graph on n vertices, numbered from 1 to n, which initially has no edges.

You have to assign each vertex an integer; let a_i be the integer on the vertex i. All a_i should be distinct integers from 1 to n.

After assigning integers, for every pair of vertices (i, j), you add an edge between them if $|i - j| + |a_i - a_j| \le k$.

Your goal is to create a graph which can be partitioned into the minimum possible (for the given values of n and k) number of cliques. Each vertex of the graph should belong to exactly one clique. Recall that a clique is a set of vertices such that every pair of vertices in it are connected with an edge.

Since BledDest hasn't really brushed his programming skills up, he can't solve the problem "given a graph, partition it into the minimum number of cliques". So we also ask you to print the partition itself.

Input

The first line contains one integer t ($1 \le t \le 1600$) — the number of test cases.

Each test case consists of one line containing two integers n and k ($2 \le n \le 40$; $1 \le k \le 2n$).

Output

For each test case, print three lines:

- the first line should contain n **distinct** integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the values you assign to the vertices;
- the second line should contain one integer q ($1 \le q \le n$) — the number of cliques you partition the graph into;
- the third line should contain n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le q$) — the partition of the graph into cliques. Where two vertices u and v are in the same clique if $c_u = c_v$.

If there are multiple answers, print any of them.

Standard Input	Standard Output
3 2 3 5 4 8 16	2 1 1 1 1 3 1 5 2 4 2 1 1 2 1 2 1 2 3 4 5 6 7 8 1 1 1 1 1 1 1 1 1

,

F. Rare Coins

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

There are n bags numbered from 1 to n, the i-th bag contains a_i golden coins and b_i silver coins.

The value of a gold coin is 1. The value of a silver coin is either 0 or 1, determined for each silver coin independently (0 with probability $\frac{1}{2}$, 1 with probability $\frac{1}{2}$).

You have to answer q independent queries. Each query is the following:

- $l\ r$ — calculate the probability that the total value of coins in bags from l to r is strictly greater than the total value in all other bags.

Input

The first line contains two integers n and q ($1 \le n, q \le 3 \cdot 10^5$) — the number of bags and the number of queries, respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^6$) — the number of gold coins in the i-th bag.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 10^6$) — the number of silver coins in the i-th bag.

Next q lines contain queries. The j-th of the next q lines contains two integers l_j and r_j ($1 \le l_j \le r_j \le n$) — the description of the j-th query.

Additional constraints on the input:

- the sum of the array a doesn't exceed 10^6;
- the sum of the array b doesn't exceed 10^6.

Output

For each query, print one integer — the probability that the total value of coins in bags from l to r is strictly greater than the total value in all other bags, taken modulo 998244353.

Formally, the probability can be expressed as an irreducible fraction $\frac{x}{y}$. You have to print the value of $x \cdot y^{-1} \bmod 998244353$, where y^{-1} is an integer such that $y \cdot y^{-1} \bmod 998244353 = 1$.

Standard Input	Standard Output
2 2 1 0 0 2 2 2 1 1	748683265 748683265
4 3 2 3 4 5 1 0 7 3	997756929 273932289 1

3 3 2 3 1 4	

Note

In both queries from the first example, the answer is $\frac{1}{4}$.

,

G. MST with Matching

Input file: standard input
Output file: standard output
Time limit: 6 seconds
Memory limit: 512 megabytes

You are given an undirected connected graph on n vertices. Each edge of this graph has a weight; the weight of the edge connecting vertices i and j is $w_{i,j}$ (or $w_{i,j} = 0$ if there is no edge between i and j). All weights are positive integers.

You are also given a positive integer c.

You have to build a spanning tree of this graph; i. e. choose exactly $(n - 1)$ edges of this graph in such a way that every vertex can be reached from every other vertex by traversing some of the chosen edges. The cost of the spanning tree is the sum of two values:

- the sum of weights of all chosen edges;
- the maximum matching in the spanning tree (i. e. the maximum size of a set of edges such that they all belong to the chosen spanning tree, and no vertex has more than one incident edge in this set), multiplied by the given integer c.

Find any spanning tree with the minimum cost. Since the graph is connected, there exists at least one spanning tree.

Input

The first line contains two integers n and c ($2 \le n \le 20$; $1 \le c \le 10^6$).

Then n lines follow. The i-th of them contains n integers $w_{i,1}, w_{i,2}, \ldots, w_{i,n}$ ($0 \le w_{i,j} \le 10^6$), where $w_{i,j}$ denotes the weight of the edge between i and j (or $w_{i,j} = 0$ if there is no such edge).

Additional constraints on the input:

- for every $i \in [1, n]$, $w_{i,i} = 0$;
- for every pair of integers (i, j) such that $i \in [1, n]$ and $j \in [1, n]$, $w_{i,j} = w_{j,i}$;
- the given graph is connected.

Output

Print one integer — the minimum cost of a spanning tree of the given graph.

Standard Input	Standard Output
4 10 0 1 8 0 1 0 1 0 8 1 0 2 0 0 2 0	21
4 5 0 1 8 0 1 0 1 0 8 1 0 2	14

0 0 2 0	

Note

In the first example, the minimum cost spanning tree consists of edges $(1, 3)$, $(2, 3)$ and $(3, 4)$. The maximum matching for it is 1.

In the second example, the minimum cost spanning tree consists of edges $(1, 2)$, $(2, 3)$ and $(3, 4)$. The maximum matching for it is 2.