Codeforces Round #527 (Div. 3)

A. Uniform String

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

You are given two integers n and k.

Your task is to construct such a string s of length n that for each i from 1 to k there is at least one i-th letter of the Latin alphabet in this string (the first letter is 'a', the second is 'b' and so on) and there are no other letters except these. You have to **maximize the minimal frequency** of some letter (the frequency of a letter is the number of occurrences of this letter in a string). If there are several possible answers, you can print **any**.

You have to answer t **independent** queries.

Input

The first line of the input contains one integer t ($1 \le t \le 100$) — the number of queries.

The next t lines are contain queries, one per line. The i-th line contains two integers n_i and k_i ($1 \le n_i \le 100, 1 \le k_i \le min(n_i, 26)$) — the length of the string in the i-th query and the number of characters in the i-th query.

Output

Print t lines. In the i-th line print the answer to the i-th query: **any** string s_i satisfying the conditions in the problem statement with constraints from the i-th query.

Example

input
3 7 3 4 4 6 2
output
cbcacab abcd baabab

Note

In the first example query the maximum possible minimal frequency is 2, it can be easily seen that the better answer doesn't exist. Other examples of correct answers: "cbcabba", "ccbbaaa" (any permutation of given answers is also correct).

In the second example query any permutation of first four letters is acceptable (the maximum minimal frequency is 1).

In the third example query any permutation of the given answer is acceptable (the maximum minimal frequency is 3).

B. Teams Forming

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

There are n students in a university. The number of students is even. The i-th student has programming skill equal to a_i.

The coach wants to form $\frac{n}{2}$ teams. Each team should consist of exactly two students, and each student should belong to exactly one team. Two students can form a team only if their skills are equal (otherwise they cannot understand each other and cannot form a team).

Students can solve problems to increase their skill. One solved problem increases the skill by one.

The coach wants to know the minimum total number of problems students should solve to form exactly $\frac{n}{2}$ teams (i.e. each pair of students should form a team). Your task is to find this number.

Input

The first line of the input contains one integer n ($2 \le n \le 100$) — the number of students. It is guaranteed that n is even.

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100$), where a_i is the skill of the i-th student.

Output

Print one number — the minimum total number of problems students should solve to form exactly $\frac{n}{2}$ teams.

Examples

input
6 5 10 2 3 14 5
output
5

input
2 1 100
output
99

Note

In the first example the optimal teams will be: $(3, 4)$, $(1, 6)$ and $(2, 5)$, where numbers in brackets are indices of students. Then, to form the first team the third student should solve 1 problem, to form the second team nobody needs to solve problems and to form the third team the second student should solve 4 problems so the answer is $1 + 4 = 5$.

In the second example the first student should solve 99 problems to form a team with the second one.

C. Prefixes and Suffixes

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

Ivan wants to play a game with you. He picked some string s of length n consisting only of lowercase Latin letters.

You don't know this string. Ivan has informed you about all its improper prefixes and suffixes (i.e. prefixes and suffixes of lengths from 1 to $n - 1$), but he didn't tell you which strings are prefixes and which are suffixes.

Ivan wants you to guess which of the given $2n - 2$ strings are prefixes of the given string and which are suffixes. It may be impossible to guess the string Ivan picked (since multiple strings may give the same set of suffixes and prefixes), but Ivan will accept your answer if there is at least one string that is consistent with it. Let the game begin!

Input

The first line of the input contains one integer number n ($2 \le n \le 100$) — the length of the guessed string s.

The next $2n - 2$ lines are contain prefixes and suffixes, one per line. Each of them is the string of length from 1 to $n - 1$ consisting only of lowercase Latin letters. They can be given in arbitrary order.

It is guaranteed that there are exactly 2 strings of each length from 1 to $n - 1$. It is also guaranteed that these strings are prefixes and suffixes of some existing string of length n.

Output

Print one string of length $2n - 2$ — the string consisting only of characters 'P' and 'S'. The number of characters 'P' should be equal to the number of characters 'S'. The i-th character of this string should be 'P' if the i-th of the input strings is the prefix and 'S' otherwise.

If there are several possible answers, you can print **any**.

Examples

input
5 ba a abab a aba baba ab aba
output
SPPSPSPS

input
3 a aa aa a

output
PPSS

input
2 a c
output
PS

Note
The only string which Ivan can guess in the first example is "ababa".

The only string which Ivan can guess in the second example is "aaa". Answers "SPSP", "SSPP" and "PSPS" are also acceptable.

In the third example Ivan can guess the string "ac" or the string "ca". The answer "SP" is also acceptable.

D1. Great Vova Wall (Version 1)

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

Vova's family is building the Great Vova Wall (named by Vova himself). Vova's parents, grandparents, grand-grandparents contributed to it. Now it's totally up to Vova to put the finishing touches.

The current state of the wall can be respresented by a sequence a of n integers, with a_i being the height of the i-th part of the wall.

Vova can only use 2×1 bricks to put in the wall (he has infinite supply of them, however).

Vova can put bricks **horizontally** on the neighboring parts of the wall of equal height. It means that if for some i the current height of part i is the same as for part $i + 1$, then Vova can put a brick there and thus increase both heights by 1. Obviously, Vova can't put bricks in such a way that its parts turn out to be off the borders (to the left of part 1 of the wall or to the right of part n of it).

The next paragraph is specific to the version 1 of the problem.

Vova can also put bricks vertically. That means increasing height of any part of the wall by 2.

Vova is a perfectionist, so he considers the wall completed when:

- all parts of the wall has the same height;
- the wall has no empty spaces inside it.

Can Vova complete the wall using any amount of bricks (possibly zero)?

Input
The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of parts in the wall.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the initial heights of the parts of the wall.

Output
Print "YES" if Vova can complete the wall using any amount of bricks (possibly zero).

Print "NO" otherwise.

Examples

input
5 2 1 1 2 5
output
YES

input
3 4 5 3
output
YES

input
2 10 10
output

YES

input
3 1 2 3
output
NO

Note

In the first example Vova can put a brick on parts 2 and 3 to make the wall $[2, 2, 2, 2, 5]$ and then put 3 bricks on parts 1 and 2 and 3 bricks on parts 3 and 4 to make it $[5, 5, 5, 5, 5]$.

In the second example Vova can put a brick vertically on part 3 to make the wall $[4, 5, 5]$, then horizontally on parts 2 and 3 to make it $[4, 6, 6]$ and then vertically on part 1 to make it $[6, 6, 6]$.

In the third example the wall is already complete.

D2. Great Vova Wall (Version 2)

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

Vova's family is building the Great Vova Wall (named by Vova himself). Vova's parents, grandparents, grand-grandparents contributed to it. Now it's totally up to Vova to put the finishing touches.

The current state of the wall can be respresented by a sequence a of n integers, with a_i being the height of the i-th part of the wall.

Vova can only use 2×1 bricks to put in the wall (he has infinite supply of them, however).

Vova can put bricks **only horizontally** on the neighbouring parts of the wall of equal height. It means that if for some i the current height of part i is the same as for part $i + 1$, then Vova can put a brick there and thus increase both heights by 1. Obviously, Vova can't put bricks in such a way that its parts turn out to be off the borders (to the left of part 1 of the wall or to the right of part n of it).

Note that Vova can't put bricks vertically.

Vova is a perfectionist, so he considers the wall completed when:

- all parts of the wall has the same height;
- the wall has no empty spaces inside it.

Can Vova complete the wall using any amount of bricks (possibly zero)?

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of parts in the wall.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the initial heights of the parts of the wall.

Output

Print "YES" if Vova can complete the wall using any amount of bricks (possibly zero).

Print "NO" otherwise.

Examples

input
5 2 1 1 2 5
output
YES

input
3 4 5 3
output
NO

input
2 10 10
output

YES

Note
In the first example Vova can put a brick on parts 2 and 3 to make the wall $[2, 2, 2, 2, 5]$ and then put 3 bricks on parts 1 and 2 and 3 bricks on parts 3 and 4 to make it $[5, 5, 5, 5, 5]$.

In the second example Vova can put no bricks in the wall.

In the third example the wall is already complete.

E. Minimal Diameter Forest

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

You are given a forest — an undirected graph with n vertices such that each its connected component is a tree.

The diameter (aka "longest shortest path") of a connected undirected graph is the maximum number of edges in the **shortest** path between any pair of its vertices.

You task is to add some edges (possibly zero) to the graph so that it becomes a tree and the diameter of the tree is minimal possible.

If there are multiple correct answers, print any of them.

Input
The first line contains two integers n and m ($1 \le n \le 1000$, $0 \le m \le n - 1$) — the number of vertices of the graph and the number of edges, respectively.

Each of the next m lines contains two integers v and u ($1 \le v, u \le n$, $v \ne u$) — the descriptions of the edges.

It is guaranteed that the given graph is a forest.

Output
In the first line print the diameter of the resulting tree.

Each of the next $(n - 1) - m$ lines should contain two integers v and u ($1 \le v, u \le n$, $v \ne u$) — the descriptions of the **added edges**.

The resulting graph should be a tree and its diameter should be minimal possible.

For $m = n - 1$ no edges are added, thus the output consists of a single integer — diameter of the given tree.

If there are multiple correct answers, print any of them.

Examples

input
4 2 1 2 2 3
output
2 4 2

input
2 0
output
1 1 2

input
3 2 1 3 2 3
output
2

Note
In the first example adding edges (1, 4) or (3, 4) will lead to a total diameter of 3. Adding edge (2, 4), however, will make it 2.

Edge (1, 2) is the only option you have for the second example. The diameter is 1.

You can't add any edges in the third example. The diameter is already 2.

F. Tree with Maximum Cost

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

You are given a tree consisting exactly of n vertices. Tree is a connected undirected graph with $n-1$ edges. Each vertex v of this tree has a value a_v assigned to it.

Let $dist(x, y)$ be the distance between the vertices x and y. The distance between the vertices is the number of edges on the simple path between them.

Let's define the cost of the tree as the following value: firstly, let's fix some vertex of the tree. Let it be v. Then the cost of the tree is $\sum_{i=1}^{n} dist(i, v) \cdot a_i$.

Your task is to calculate the **maximum possible cost** of the tree if you can choose v arbitrarily.

Input

The first line contains one integer n, the number of vertices in the tree ($1 \le n \le 2 \cdot 10^5$).

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 2 \cdot 10^5$), where a_i is the value of the vertex i.

Each of the next $n-1$ lines describes an edge of the tree. Edge i is denoted by two integers u_i and v_i, the labels of vertices it connects ($1 \le u_i, v_i \le n$, $u_i \ne v_i$).

It is guaranteed that the given edges form a tree.

Output

Print one integer — the **maximum possible cost** of the tree if you can choose any vertex as v.

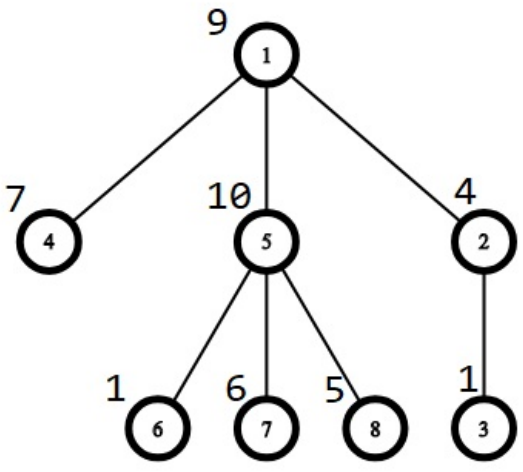
Examples

input
8 9 4 1 7 10 1 6 5 1 2 2 3 1 4 1 5 5 6 5 7 5 8
output
121

input
1 1337
output
0

Note

Picture corresponding to the first example:



You can choose the vertex 3 as a root, then the answer will be
$2 \cdot 9 + 1 \cdot 4 + 0 \cdot 1 + 3 \cdot 7 + 3 \cdot 10 + 4 \cdot 1 + 4 \cdot 6 + 4 \cdot 5 = 18 + 4 + 0 + 21 + 30 + 4 + 24 + 20 = 121$.

In the second example tree consists only of one vertex so the answer is always 0.
