A. Little Artem

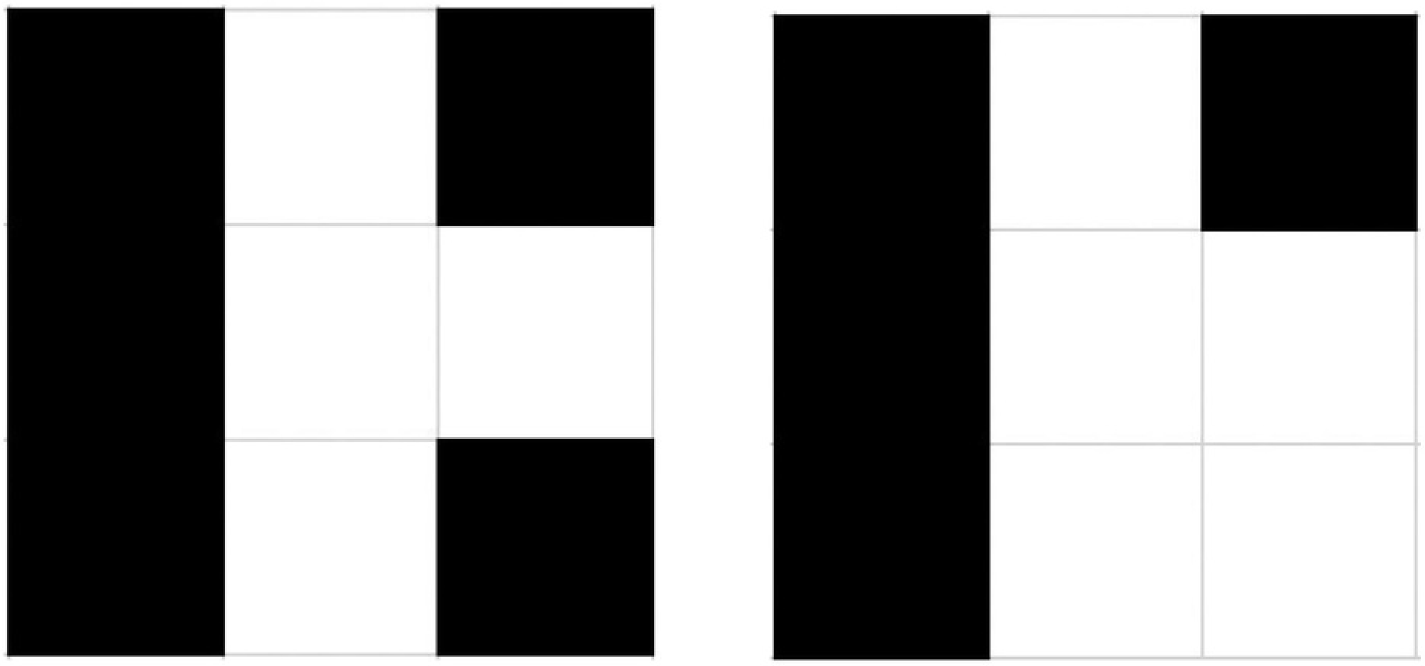
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Young boy Artem tries to paint a picture, and he asks his mother Medina to help him. Medina is very busy, that's why she asked for your help.

Artem wants to paint an $n \times m$ board. Each cell of the board should be colored in white or black.

Lets B be the number of black cells that have at least one white neighbor adjacent by the side. Let W be the number of white cells that have at least one black neighbor adjacent by the side. A coloring is called **good** if $B = W + 1$.

The first coloring shown below has $B = 5$ and $W = 4$ (all cells have at least one neighbor with the opposite color). However, the second coloring is not **good** as it has $B = 4$, $W = 4$ (only the bottom right cell doesn't have a neighbor with the opposite color).



Please, help Medina to find any **good** coloring. It's guaranteed that under given constraints the solution always exists. If there are several solutions, output any of them.

Input

Each test contains multiple test cases.

The first line contains the number of test cases t ($1 \le t \le 20$). Each of the next t lines contains two integers n, m ($2 \le n, m \le 100$) — the number of rows and the number of columns in the grid.

Output

For each test case print n lines, each of length m, where i-th line is the i-th row of your colored matrix (cell labeled with `'B'` means that the cell is black, and `'W'` means white). Do not use quotes.

It's guaranteed that under given constraints the solution always exists.

Example

input

```
2
3 2
3 3
```

output

```
BW
WB
BB
BWB
BWW
BWB
```

Note

In the first testcase, $B = 3$, $W = 2$.

In the second testcase, $B = 5$, $W = 4$. You can see the coloring in the statement.

B. Kind Anton

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

Once again, Boris needs the help of Anton in creating a task. This time Anton needs to solve the following problem:

There are two arrays of integers a and b of length n. It turned out that array a contains only elements from the set $\{-1, 0, 1\}$.

Anton can perform the following sequence of operations any number of times:

1. Choose any pair of indexes (i, j) such that $1 \le i < j \le n$. It is possible to choose the same pair (i, j) more than once.
2. Add a_i to a_j. In other words, j-th element of the array becomes equal to $a_i + a_j$.

For example, if you are given array $[1, -1, 0]$, you can transform it only to $[1, -1, -1]$, $[1, 0, 0]$ and $[1, -1, 1]$ by one operation.

Anton wants to predict if it is possible to apply some number (zero or more) of these operations to the array a so that it becomes equal to array b. Can you help him?

Input

Each test contains multiple test cases.

The first line contains the number of test cases t ($1 \le t \le 10000$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the length of arrays.

The second line of each test case contains n integers $a_1, a_2, \dots, a_n$ ($-1 \le a_i \le 1$) — elements of array a. There can be duplicates among elements.

The third line of each test case contains n integers $b_1, b_2, \dots, b_n$ ($-10^9 \le b_i \le 10^9$) — elements of array b. There can be duplicates among elements.

It is guaranteed that the sum of n over all test cases doesn't exceed 10^5.

Output

For each test case, output one line containing "YES" if it's possible to make arrays a and b equal by performing the described operations, or "NO" if it's impossible.

You can print each letter in any case (upper or lower).

Example

input

```
5
3
1 -1 0
1 1 -2
3
0 1 1
0 2 2
2
1 0
1 41
2
-1 0
-1 -41
5
0 1 -1 1 -1
1 1 -1 1 -1
```

output

```
YES
NO
YES
YES
NO
```

Note

In the first test-case we can choose $(i, j) = (2, 3)$ twice and after that choose $(i, j) = (1, 2)$ twice too. These operations will transform $[1, -1, 0] \to [1, -1, -2] \to [1, 1, -2]$

In the second test case we can't make equal numbers on the second position.

In the third test case we can choose $(i, j) = (1, 2)$ 41 times. The same about the fourth test case.

In the last lest case, it is impossible to make array a equal to the array b.

C. Eugene and an array

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Eugene likes working with arrays. And today he needs your help in solving one challenging task.

An array c is a subarray of an array b if c can be obtained from b by deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

Let's call a nonempty array **good** if for every nonempty subarray of this array, sum of the elements of this subarray is nonzero. For example, array $[-1, 2, -3]$ is **good**, as all arrays $[-1]$, $[-1, 2]$, $[-1, 2, -3]$, $[2]$, $[2, -3]$, $[-3]$ have nonzero sums of elements. However, array $[-1, 2, -1, -3]$ isn't **good**, as his subarray $[-1, 2, -1]$ has sum of elements equal to 0.

Help Eugene to calculate the number of nonempty **good** subarrays of a given array a.

Input

The first line of the input contains a single integer n $(1 \le n \le 2 \times 10^5)$ — the length of array a.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ $(-10^9 \le a_i \le 10^9)$ — the elements of a.

Output

Output a single integer — the number of **good** subarrays of a.

Examples

input

```
3
1 2 -3
```

output

```
5
```

input

```
3
41 -41 41
```

output

```
3
```

Note

In the first sample, the following subarrays are **good**: $[1]$, $[1, 2]$, $[2]$, $[2, -3]$, $[-3]$. However, the subarray $[1, 2, -3]$ isn't **good**, as its subarray $[1, 2, -3]$ has sum of elements equal to 0.

In the second sample, three subarrays of size 1 are the only **good** subarrays. At the same time, the subarray $[41, -41, 41]$ isn't **good**, as its subarray $[41, -41]$ has sum of elements equal to 0.

D. Challenges in school №41

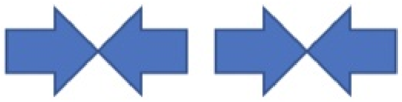
time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n children, who study at the school №41. It is well-known that they are good mathematicians. Once at a break, they arranged a challenge for themselves. All children arranged in a row and turned heads either to the left or to the right.

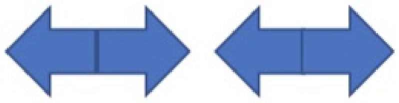
Children can do the following: in one second several pairs of neighboring children who are **looking at each other** can **simultaneously** turn the head in the opposite direction. For instance, the one who was looking at the right neighbor turns left and vice versa for the second child. Moreover, every second **at least one** pair of neighboring children performs such action. They are going to finish when there is no pair of neighboring children who are looking at each other.

You are given the number n, the initial arrangement of children and the number k. You have to find a way for the children to act if they want to finish the process in exactly k seconds. More formally, for each of the k moves, you need to output the numbers of the children who turn left during this move.

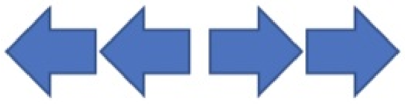
For instance, for the configuration shown below and $k = 2$ children can do the following steps:



At the beginning, two pairs make move: $(1, 2)$ and $(3, 4)$. After that, we receive the following configuration:



At the second move pair $(2, 3)$ makes the move. The final configuration is reached. Good job.



It is guaranteed that if the solution exists, it takes not more than n^2 "headturns".

Input

The first line of input contains two integers n and k ($2 \le n \le 3000$, $1 \le k \le 3000000$) — the number of children and required number of moves.

The next line contains a string of length n and consists only of characters L and R, where L means that the child looks to the left and R means that the child looks to the right.

Output

If there is no solution, print a single line with number -1.

Otherwise, output k lines. Each line has to start with a number n_i ($1 \le n_i \le \frac{n}{2}$) — the number of pairs of children, who turn at this move. After that print n_i **distinct** integers — the numbers of the children who will turn left during this move.

After performing all "headturns", there can't be a pair of two neighboring children looking at each other.

If there are many solutions, print any of them.

Examples

input

```
2 1
RL
```

output

```
1 1
```

input

```
2 1
LR
```

output

```
-1
```

input

```
4 2
RLRL
```

output

```
2 1 3
1 2
```

Note

The first sample contains a pair of children who look at each other. After one move, they can finish the process.

In the second sample, children can't make any move. As a result, they can't end in $k > 0$ moves.

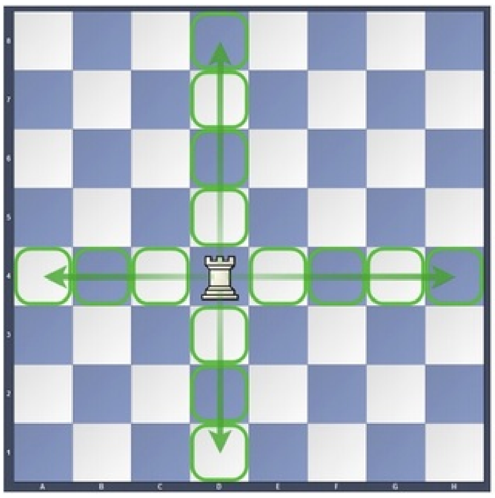
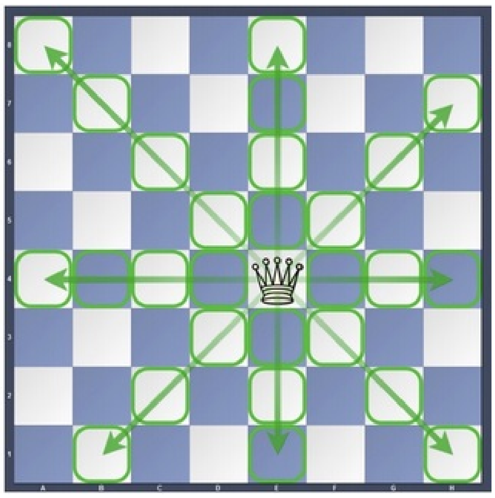
The third configuration is described in the statement.

E. Road to 1600

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

Egor wants to achieve a rating of 1600 points on the well-known chess portal ChessForces and he needs your help!

Before you start solving the problem, Egor wants to remind you how the chess pieces move. Chess **rook** moves along straight lines up and down, left and right, as many squares as it wants. And when it wants, it can stop. The **queen** walks in all directions vertically and diagonally at any distance. You can see the examples below.

To reach the goal, Egor should research the next topic:

There is an $N \times N$ board. Each cell of the board has a number from 1 to N^2 in it and numbers in all cells are distinct.

In the beginning, some chess figure stands in the cell with the number 1. Note that this cell is already considered as visited. After that every move is determined by the following rules:

1. Among all **not visited** yet cells to which the figure can get in one move, it goes to the cell that has **minimal** number.
2. If all accessible cells were already visited and some cells are not yet visited, then the figure is teleported to the not visited cell that has minimal number. If this step happens, the piece pays a fee of 1 **vun**.
3. If all cells are already visited, the process is stopped.

Egor should find an $N \times N$ board on which the rook pays **strictly less vuns** than the queen during the round with this numbering. Help him to find such $N \times N$ numbered board, or tell that it doesn't exist.

Input
The only line contains one integer N — the size of the board, $1 \le N \le 500$.

Output
The output should contain N lines.

In i-th line output N numbers — numbers on the i-th row of the board. All numbers from 1 to $N \times N$ must be used exactly once.

On your board rook must pay strictly less vuns than the queen.

If there are no solutions, print -1.

If there are several solutions, you can output any of them.

Examples

input
```
1
```
output
```
-1
```

input
```
4
```
output
```
4 3 6 12
7 5 9 15
14 1 11 10
13 8 16 2
```

Note
In case we have 1×1 board, both rook and queen do not have a chance to pay fees.

In second sample **rook** goes through cells
$1 \to 3 \to 4 \to 6 \to 9 \to 5 \to 7 \to 13 \to 2 \to 8 \to 16 \to 11 \to 10 \to 12 \to 15 \to \mathbf{(1\ vun)} \to 14.$

Queen goes through
$1 \to 3 \to 4 \to 2 \to 5 \to 6 \to 9 \to 7 \to 13 \to 8 \to 11 \to 10 \to 12 \to 15 \to \mathbf{(1\ vun)} \to 14 \to \mathbf{(1\ vun)} \to 16.$

As a result rook pays 1 vun and queen pays 2 vuns.

F. Kate and imperfection

time limit per test: 1 second
memory limit per test: 256 megabytes

input: standard input
output: standard output

Kate has a set S of n integers $\{1, \dots, n\}$.

She thinks that **imperfection** of a subset $M \subseteq S$ is equal to the **maximum** of $gcd(a, b)$ over all pairs (a, b) such that both a and b are in M and $a \neq b$.

Kate is a very neat girl and for each $k \in \{2, \dots, n\}$ she wants to find a subset that has the **smallest imperfection** among all subsets in S of size k. There can be more than one subset with the smallest imperfection and the same size, but you don't need to worry about it. Kate wants to find all the subsets herself, but she needs your help to find the smallest possible imperfection for each size k, will name it I_k.

Please, help Kate to find I_2, I_3, ..., I_n.

Input

The first and only line in the input consists of only one integer n ($2 \le n \le 5 \cdot 10^5$) — the size of the given set S.

Output

Output contains only one line that includes $n - 1$ integers: I_2, I_3, ..., I_n.

Examples

input

```
2
```

output

```
1
```

input

```
3
```

output

```
1 1
```

Note

First sample: answer is 1, because $gcd(1, 2) = 1$.

Second sample: there are subsets of S with sizes $2, 3$ with imperfection equal to 1. For example, $\{2, 3\}$ and $\{1, 2, 3\}$.