

**Codeforces Round #806 (Div. 4)****A. YES or YES?**

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

There is a string  $s$  of length 3, consisting of uppercase and lowercase English letters. Check if it is equal to "YES" (without quotes), where each letter can be in any case. For example, "yES", "Yes", "yes" are all allowable.

**Input**

The first line of the input contains an integer  $t$  ( $1 \leq t \leq 10^3$ ) — the number of testcases.

The description of each test consists of one line containing one string  $s$  consisting of three characters. Each character of  $s$  is either an uppercase or lowercase English letter.

**Output**

For each test case, output "YES" (without quotes) if  $s$  satisfies the condition, and "NO" (without quotes) otherwise.

You can output "YES" and "NO" in any case (for example, strings "yES", "yes" and "Yes" will be recognized as a positive response).

**Example**

input
10 YES yES yes Yes YeS Noo orZ yEz Yas XES
output
YES YES YES YES YES NO NO NO NO NO

**Note**

The first five test cases contain the strings "YES", "yES", "yes", "Yes", "YeS". All of these are equal to "YES", where each character is either uppercase or lowercase.

**B. ICPC Balloons**

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

In an ICPC contest, balloons are distributed as follows:

- Whenever a team solves a problem, that team gets a balloon.
- The first team to solve a problem gets an additional balloon.

A contest has 26 problems, labelled A, B, C, ..., Z. You are given the order of solved problems in the contest, denoted as a string  $s$ , where the  $i$ -th character indicates that the problem  $s_i$  has been solved by some team. No team will solve the same problem twice. Determine the total number of balloons that the teams received. Note that some problems may be solved by none of the teams.

**Input**

The first line of the input contains an integer  $t$  ( $1 \leq t \leq 100$ ) — the number of testcases.

The first line of each test case contains an integer  $n$  ( $1 \leq n \leq 50$ ) — the length of the string.

The second line of each test case contains a string  $s$  of length  $n$  consisting of uppercase English letters, denoting the order of solved problems.

**Output**

For each test case, output a single integer — the total number of balloons that the teams received.

### Example

input
6 3 ABA 1 A 3 ORZ 5 BAAAA 4 BKPT 10 CODEFORCES
output
5 2 6 7 8 17

### Note

In the first test case, 5 balloons are given out:

- Problem A is solved. That team receives 2 balloons: one because they solved the problem, an an additional one because they are the first team to solve problem A.
- Problem B is solved. That team receives 2 balloons: one because they solved the problem, an an additional one because they are the first team to solve problem B.
- Problem A is solved. That team receives only 1 balloon, because they solved the problem. Note that they don't get an additional balloon because they are **not** the first team to solve problem A.

The total number of balloons given out is  $2 + 2 + 1 = 5$ .

In the second test case, there is only one problem solved. The team who solved it receives 2 balloons: one because they solved the problem, an an additional one because they are the first team to solve problem A.

## C. Cypher

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

Luca has a cypher made up of a sequence of  $n$  wheels, each with a digit  $a_i$  written on it. On the  $i$ -th wheel, he made  $b_i$  moves. Each move is one of two types:

- *up* move (denoted by  $\mathsf{U}$ ): it increases the  $i$ -th digit by 1. After applying the up move on 9, it becomes 0.
- *down* move (denoted by  $\mathsf{D}$ ): it decreases the  $i$ -th digit by 1. After applying the down move on 0, it becomes 9.



Example for  $n = 4$ . The current sequence is 0 0 0 0.

Luca knows the final sequence of wheels and the moves for each wheel. Help him find the original sequence and crack the cypher.

### Input

The first line contains a single integer  $t$  ( $1 \leq t \leq 100$ ) — the number of test cases.

The first line of each test case contains a single integer  $n$  ( $1 \leq n \leq 100$ ) — the number of wheels.

The second line contains  $n$  integers  $a_i$  ( $0 \leq a_i \leq 9$ ) — the digit shown on the  $i$ -th wheel after all moves have been performed.

Then  $n$  lines follow, the  $i$ -th of which contains the integer  $b_i$  ( $1 \leq b_i \leq 10$ ) and  $b_i$  characters that are either  $\mathsf{U}$  or  $\mathsf{D}$  — the number of moves performed on the  $i$ -th wheel, and the moves performed.  $\mathsf{U}$  and  $\mathsf{D}$  represent an *up* move and a *down* move respectively.

### Output

For each test case, output  $n$  space-separated digits — the initial sequence of the cypher.

### Example

input
-------

3  
3  
9 3 1  
3 DDD  
4 UDUU  
2 DU  
2  
0 9  
9 DDDDDDDDD  
9 UUUUUUUUU  
5  
0 5 9 8 3  
10 UUUUUUUUUU  
3 UUD  
8 UUDUUDDDD  
10 UUDUUDDUDDU  
4 UUUU

output

2 1 1  
9 0  
0 4 9 6 9

Note

In the first test case, we can prove that initial sequence was [2, 1, 1]. In that case, the following moves were performed:

- On the first wheel:  $2 \xrightarrow{D} 1 \xrightarrow{D} 0 \xrightarrow{D} 9$ .
- On the second wheel:  $1 \xrightarrow{U} 2 \xrightarrow{D} 1 \xrightarrow{U} 2 \xrightarrow{U} 3$ .
- On the third wheel:  $1 \xrightarrow{D} 0 \xrightarrow{U} 1$ .

The final sequence was [9, 3, 1], which matches the input.

D. Double Strings

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

You are given  $n$  strings  $s_1, s_2, \dots, s_n$  of length at most 8.

For each string  $s_i$ , determine if there exist two strings  $s_j$  and  $s_k$  such that  $s_i = s_j + s_k$ . That is,  $s_i$  is the concatenation of  $s_j$  and  $s_k$ . Note that  $j$  can be equal to  $k$ .

Recall that the concatenation of strings  $s$  and  $t$  is  $s + t = s_1s_2 \dots s_pt_1t_2 \dots t_q$ , where  $p$  and  $q$  are the lengths of strings  $s$  and  $t$  respectively. For example, concatenation of "code" and "forces" is "codeforces".

Input

The first line contains a single integer  $t$  ( $1 \leq t \leq 10^4$ ) — the number of test cases.

The first line of each test case contains a single integer  $n$  ( $1 \leq n \leq 10^5$ ) — the number of strings.

Then  $n$  lines follow, the  $i$ -th of which contains non-empty string  $s_i$  of length at most 8, consisting of lowercase English letters. Among the given  $n$  strings, there may be equal (duplicates).

The sum of  $n$  over all test cases doesn't exceed  $10^5$ .

Output

For each test case, output a binary string of length  $n$ . The  $i$ -th bit should be 1 if there exist two strings  $s_j$  and  $s_k$  where  $s_i = s_j + s_k$ , and 0 otherwise. Note that  $j$  can be equal to  $k$ .

Example

input
3 5 abab ab abc abacb c 3 x xx xxx 8 codeforc es codes cod forc forces e code
output
10100

011  
10100101

Note

In the first test case, we have the following:

- $s_1 = s_2 + s_2$ , since  $\mathbf{abab} = \mathbf{ab} + \mathbf{ab}$ . Remember that  $j$  can be equal to  $k$ .
- $s_2$  is not the concatenation of any two strings in the list.
- $s_3 = s_2 + s_5$ , since  $\mathbf{abc} = \mathbf{ab} + \mathbf{c}$ .
- $s_4$  is not the concatenation of any two strings in the list.
- $s_5$  is not the concatenation of any two strings in the list.

Since only  $s_1$  and  $s_3$  satisfy the conditions, only the first and third bits in the answer should be 1, so the answer is 10100.

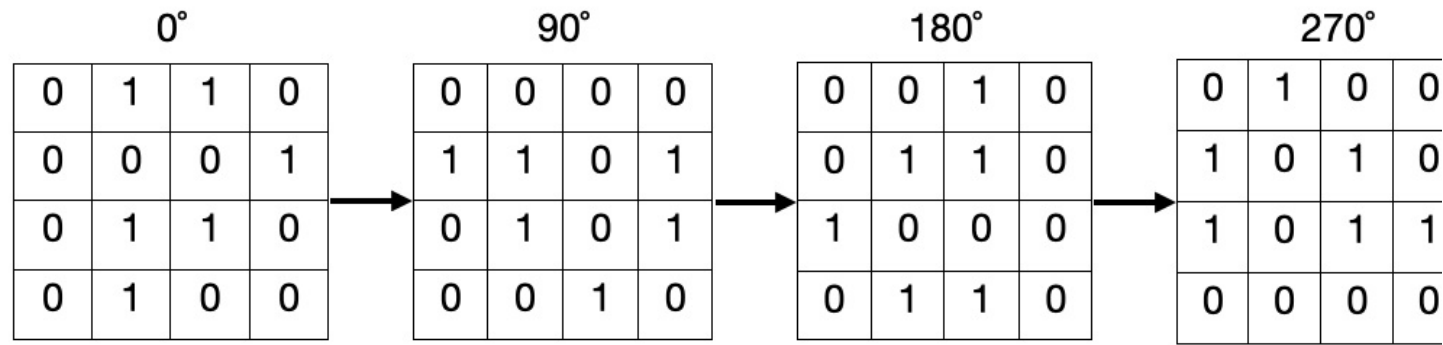
E. Mirror Grid

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

You are given a square grid with  $n$  rows and  $n$  columns. Each cell contains either 0 or 1.

In an operation, you can select a cell of the grid and flip it (from  $0 \rightarrow 1$  or  $1 \rightarrow 0$ ). Find the minimum number of operations you need to obtain a square that remains the same when rotated  $0^\circ$ ,  $90^\circ$ ,  $180^\circ$  and  $270^\circ$ .

The picture below shows an example of all rotations of a grid.



Input

The first line contains a single integer  $t$  ( $1 \leq t \leq 100$ ) — the number of test cases.

The first line of each test case contains a single integer  $n$  ( $1 \leq n \leq 100$ ) — the size of the grid.

Then  $n$  lines follow, each with  $n$  characters  $a_{i,j}$  ( $0 \leq a_{i,j} \leq 1$ ) — the number written in each cell.

Output

For each test case output a single integer — the minimum number of operations needed to make the square look the same rotated  $0^\circ$ ,  $90^\circ$ ,  $180^\circ$  and  $270^\circ$ .

Example

input
5 3 010 110 010 1 0 5 11100 11011 01011 10011 11000 5 01000 10101 01010 00010 01001 5 11001 00000 11111 10110 01111
output
1 0 9 7 6

**Note**

0 1 0

In the first test case, we can perform one operations to make the grid  $\begin{matrix} 1 & 1 & 1 \\ 0 & 1 & 0 \end{matrix}$ . Now, all rotations of the square are the same.

0 1 0

In the second test case, all rotations of the square are already the same, so we don't need any flips.

**F. Yet Another Problem About Pairs Satisfying an Inequality**

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given an array  $a_1, a_2, \dots, a_n$ . Count the number of pairs of indices  $1 \leq i, j \leq n$  such that  $a_i < i < a_j < j$ .

**Input**

The first line contains an integer  $t$  ( $1 \leq t \leq 1000$ ) — the number of test cases.

The first line of each test case contains an integer  $n$  ( $2 \leq n \leq 2 \cdot 10^5$ ) — the length of the array.

The second line of each test case contains  $n$  integers  $a_1, a_2, \dots, a_n$  ( $0 \leq a_i \leq 10^9$ ) — the elements of the array.

It is guaranteed that the sum of  $n$  across all test cases does not exceed  $2 \cdot 10^5$ .

**Output**

For each test case, output a single integer — the number of pairs of indices satisfying the condition in the statement.

Please note, that the answer for some test cases won't fit into 32-bit integer type, so you should use at least 64-bit integer type in your programming language (like `long long` for C++).

**Example**

input
5 8 1 1 2 3 8 2 1 4 2 1 2 10 0 2 1 6 3 4 1 2 8 3 2 1 1000000000 3 0 1000000000 2
output
3 0 10 0 1

**Note**

For the first test cases the pairs are  $(i, j) = \{(2, 4), (2, 8), (3, 8)\}$ .

- The pair  $(2, 4)$  is true because  $a_2 = 1, a_4 = 3$  and  $1 < 2 < 3 < 4$ .
- The pair  $(2, 8)$  is true because  $a_2 = 1, a_8 = 4$  and  $1 < 2 < 4 < 8$ .
- The pair  $(3, 8)$  is true because  $a_3 = 2, a_8 = 4$  and  $2 < 3 < 4 < 8$ .

**G. Good Key, Bad Key**

time limit per test: 3 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

There are  $n$  chests. The  $i$ -th chest contains  $a_i$  coins. You need to open all  $n$  chests **in order from chest 1 to chest  $n$** .

There are two types of keys you can use to open a chest:

- a good key, which costs  $k$  coins to use;
- a bad key, which does not cost any coins, but will halve all the coins in each unopened chest, **including the chest it is about to open**. The halving operation **will round down** to the nearest integer for each chest halved. In other words using a bad key to open chest  $i$  will do  $a_i = \lfloor \frac{a_i}{2} \rfloor, a_{i+1} = \lfloor \frac{a_{i+1}}{2} \rfloor, \dots, a_n = \lfloor \frac{a_n}{2} \rfloor$ ;
- any key (both good and bad) breaks after a usage, that is, it is a one-time use.

You need to use in total  $n$  keys, one for each chest. Initially, you have no coins and no keys. If you want to use a good key, then you need to buy it.

During the process, you are allowed to go into debt; for example, if you have 1 coin, you are allowed to buy a good key worth  $k = 3$  coins, and your balance will become  $-2$  coins.

Find the maximum number of coins you can have after opening all  $n$  chests in order from chest 1 to chest  $n$ .

### Input

The first line contains a single integer  $t$  ( $1 \leq t \leq 10^4$ ) — the number of test cases.

The first line of each test case contains two integers  $n$  and  $k$  ( $1 \leq n \leq 10^5$ ;  $0 \leq k \leq 10^9$ ) — the number of chests and the cost of a good key respectively.

The second line of each test case contains  $n$  integers  $a_i$  ( $0 \leq a_i \leq 10^9$ ) — the amount of coins in each chest.

The sum of  $n$  over all test cases does not exceed  $10^5$ .

### Output

For each test case output a single integer — the maximum number of coins you can obtain after opening the chests in order from chest 1 to chest  $n$ .

Please note, that the answer for some test cases won't fit into 32-bit integer type, so you should use at least 64-bit integer type in your programming language (like `long long` for C++).

### Example

input
5 4 5 10 10 3 1 1 2 1 3 12 10 10 29 12 51 5 74 89 45 18 69 67 67 11 96 23 59 2 57 85 60
output
11 0 13 60 58

### Note

In the first test case, one possible strategy is as follows:

- Buy a good key for 5 coins, and open chest 1, receiving 10 coins. Your current balance is  $0 + 10 - 5 = 5$  coins.
- Buy a good key for 5 coins, and open chest 2, receiving 10 coins. Your current balance is  $5 + 10 - 5 = 10$  coins.
- Use a bad key and open chest 3. As a result of using a bad key, the number of coins in chest 3 becomes  $\left\lfloor \frac{3}{2} \right\rfloor = 1$ , and the number of coins in chest 4 becomes  $\left\lfloor \frac{1}{2} \right\rfloor = 0$ . Your current balance is  $10 + 1 = 11$ .
- Use a bad key and open chest 4. As a result of using a bad key, the number of coins in chest 4 becomes  $\left\lfloor \frac{0}{2} \right\rfloor = 0$ . Your current balance is  $11 + 0 = 11$ .

At the end of the process, you have 11 coins, which can be proven to be maximal.