Assignment 2

#	Title	Score
A	Polynomial Multiplication	100
B	Nethersea Land	100
C	Palindromic Substring	100
D	Urara's Divination	100 + 30 (bonus)

Problem A: Polynomial Multiplication

Description

Sora loves math. He also loves programming. So, he decided to design a program that helps him calculate math faster. The most difficult problem in calculating is the polynomial multiplication. Great, he thought, because it doubled the happiness of thinking.

If you were Sora, how will you solve this problem?

Input

The first line includes one integer T, as the number of cases.

The following $2T$ lines, two lines for each case, represent the parameters of two polynomials.

At the first line for each case $n, a_{n-1}, a_{n-2}, \cdots, a_1, a_0$ will be given.

And then $m, b_{m-1}, b_{m-2}, \cdots, b_1, b_0$ are the second line, separated by one space.

Polynomial $A = a_{n-1}x^{n-1} + \cdots + a_2x^2 + a_1x + a_0$, if the line has n integers.

Polynomial $B = b_{m-1}x^{m-1} + \cdots + b_2x^2 + b_1x + b_0$, if the line has m integers.

Beware that there may be $n \neq m$.

Sample Input

```
1
3 3 2 5
3 5 1 2
```

$$A = 3x^2 + 2x + 5$$
$$B = 5x^2 + x + 2$$

Output

Print the parameters of $A \times B$.

Sample Output

```
15 13 33 9 10
```

$$A \times B = 15x^4 + 13x^3 + 33x^2 + 9x + 10$$

Contraints

It's guaranteed that $0 < T \leq 100$, $0 < n, m \leq 100$ and $|a_i|, |b_i| \leq 100$.

Problem B: Nethersea Land

Statement

There's a 2D $n \times m$ grid map. There are three types of each cell:

- **Wall**,
- **Land**,
- Land with **Brand**.

Initially, there is only one Brand on the map. Each second, every Brand will spread to the adjacent land (not wall) on four directions.

Given a map, your task is to calculate the map after k seconds.

Input

In the first line, there are three positive integers n, m, k.

On the following n lines, each line contains m characters separated by one space. `W` means a wall, `L` means a land, and `B` means a Brand.

Output

Print the map after k seconds, the format is the same as that in input.

Sample

Input

```
5 5 3
L L L W L
W L L W L
L W L L L
L W B W L
L L W W L
```

Output

```
L L B W L
W B B W L
L W B B B
L W B W L
L L W W L
```

Contraints

It's guaranteed that $0 < n, m, k \leq 100$.

Note

Hint 1: you can scan the whole array every second and make the brands spread one grid.

The task can be accomplished without any advanced algorithms, such as BFS or DFS. If you don't know what they are, just leave it alone. It's just to tell you this problem is not beyond the syllabus.

Problem C: Palindromic Substring

Description

Your computer is out of storage. You need a compression algorithm now, right away! However, giant trees grow from seeds. You need to start from small things, for example, if a substring S of a very very long string is palindromic, it can be expressed as $S = BAB^T$ or $S = BB^T$, where A is a single character and B^T is the reverse version of B, say, $B_i = B^T_{n-i-1}$, let B_i be the i-th character of B. Through this way, the redundant data will be found and deleted, and you'll have more spare storage. :D

Your job, willingly or not, is to design a program that finds the number of substrings in a very looooooooooong string.

Input

The first line includes one integer T, as the number of cases.

The following T lines consist of the string for each case.

It's ensured that the letters are lower case.

Sample Input

```
3
abc
aaa
eabcdcba
```

Output

Print the number of palindromic substrings for each case.

Sample Output

```
3
6
11
```

Explanation

3: `a`, `b`, `c`

6: `a`, `a`, `a`, `aa`, `aa`, `aaa`

11: `e`, `a`, `b`, `c` `d`, `cdc`, `c`, `b`, `a`, `bcdcb`, `abcdcba`

If the substrings with the same character(s) appear in different position, they are considered different substrings. The minimum substring contains only one character.

Contraints

It's guaranteed that $0 < T \leq 100$ and $0 < n \leq 100$, where n is the length of strings.

Hint

You can use `.toCharArray()` to convert a `string` to a char array (`char[]`).

The length of strings will not be given, so you may use `.length()` to access its length.

Problem D: Urara's Divination

This is an interactive problem. You do NOT need to use `Scanner` for input, or print your answer using `System.out`.

Description

Chiya invite you to play a game. She comes up with an array x with length n (n is a multiple of 3) whose values are either 0 or 1. But she doesn't tell you the content of the array, and requests you to guess the array by asking several questions.

In one query (i.e. asking), you can choose three distinct positions a, b, c, and she will tell you the most common value of the three positions. That is, if there are more 1s than 0s in x_a, x_b, x_c, she will tell you "1", and "0" otherwise.

One more thing, it's **guaranteed** that $\frac{1}{3}n < m < \frac{2}{3}n$, where m is the number of 1s.

More technically, you are required to implement a function `solve` in class `Main` that accepts an argument n, and returns an array x' with length n:

```java
public class Main {
    public static int[] solve(int n) {
        // TODO: write your code here
    }
}
```

To recover this array, you can make several queries in your code by invoking `Judge.query`. For your information, its signature is

```java
public static int query(int a, int b, int c);
```

It accepts three arguments a, b, c, indicating the three indices of the array. If the number of `1`s is more than that of `0`s at x_a, x_b, x_c, it will return `1`, and `0` otherwise.

By making queries, your task is to recover an array x' such that $x' = x$, and return it.

That's all of the core part of the problem.

Template

To reduce unnecessary coding, we provide a template for you to get started and to debug.

The class `Judge` is to help you debug locally, you can execute `Judge.main` in IDEA after you finish coding. When the program runs, you will first need to enter an integer n (the length) and then an array x, so that it can act like Chiya to interactive with your code in `Main`, check whether your code is able to recover the array, and provide some useful debug information.

This class has nothing to do with the final judging on OJ, so you can modify it if you understand, if not, just leave it alone.

When submitting your code, you only need to submit your class `Main` and some necessary `import`s, do **NOT** paste the class `Judge`. That is, submit the code until the line "`DO NOT SUBMIT THE FOLLOWING CODE`".

The following template can also be downloaded from https://send.cra.moe/file/58pnoATqrkfljh6q/0xkXS7Xoxp9WTs0Q/Main.java.

```java
public class Main {
    public static int[] solve(int n) {
        // TODO: write your code here
    }
}

// !!!! DO NOT SUBMIT THE FOLLOWING CODE !!!!
class Judge {
    private static int n;
    private static int[] x;
    private static int count = 0;

    public static int query(int a, int b, int c) {
        // check duplicated indices
```

```
        if (a == b || b == c || a == c) {
            System.err.printf("[!] Duplicated indices: %d %d %d\n", a, b, c);
            System.exit(1);
        }
        // check out-of-bound
        if (a >= n || b >= n || c >= n || a < 0 || b < 0 || c < 0) {
            System.err.printf("[!] Indices out of range: %d %d %d\n", a, b, c);
            System.exit(1);
        }

        count++;  // increase the guessing times

        int sum = x[a] + x[b] + x[c];  // count the number of 1s
        System.out.printf("[+] Query %d %d %d => %d\n", a, b, c, sum / 2);
        return sum / 2;  // if there are two or more 1s, it will return 1
    }

    public static void main(String[] args) {
        java.util.Scanner scanner = new java.util.Scanner(System.in);

        System.out.print("[*] n: ");
        n = scanner.nextInt();

        // read an array x[]
        System.out.print("[*] x[]: ");
        x = new int[n];
        for (int i = 0; i < n; i++)
            x[i] = scanner.nextInt();

        // call your code
        int[] y = Main.solve(n);

        // compare your answer with the correct array
        boolean correct = true;
        for (int i = 0; i < n; i++)
            if (x[i] != y[i])
                correct = false;

        System.out.println();
        System.out.println(correct ? "[+] Correct" : "[-] Wrong");
        System.out.println("[+] Number of guesses: " + count);
        System.out.println("[+] Your answer: " + java.util.Arrays.toString(y));
        System.out.println("[+]   The array: " + java.util.Arrays.toString(x));
    }
}
```

Sample Input

This sample input is for local test only. No sample output is provided, because you can get the result from the local checker.

```
6
1 1 1 0 0 0
```

Example

Take this dumb program for example:

```java
public class Main {
    public static int[] solve(int n) {
        // "Let's assume that n = 6"

        int[] x = new int[n];           // default values for items are all 0
        if (Judge.guess(0, 1, 2) == 1)
            x[0] = x[1] = x[2] = 1;     // I'm foolish
        if (Judge.guess(3, 4, 5) == 1)
            x[3] = x[4] = x[5] = 1;     // I'm stupid
        return x;
    }
}
```

This program first queries for the first three indices $(0, 1, 2)$, if there's more 1, it just assumes that they're all 1. Then it queries for the last three indices $(3, 4, 5)$, and same as above, if there's more 1, it just assumes that they're all 1 again.

This program may not be able to correctly recover the array, but it briefly shows the procedure of this interactive problem. You can make queries freely and dynamically - the queries can be based on the current state and knowledge in your program.

Constraints

It's guaranteed that $6 \le n \le 10^4$. The return value of `Judge.query` is either 0 or 1.

Grading

Let k be the number of your guesses in one game.

You will get

- 100 points, as long as you can recover all the values (unlimited queries).
- another 10 points bonus, if you recover the array and $k \le 2n$.
- another 10 points bonus, if you recover the array and $k \le \frac{3}{2}n$.
- another 10 points bonus, if you recover the array and $k \le n + 1$.

Hints

Hint 1: How to recover one 0 and one 1 in the array?

Hint 2: For four items (a, b, c, d), if $\text{query}(a, b, c) = 0$ and $\text{query}(b, c, d) = 1$, what can you infer?