QUESTION 1: Minimise Oneness => CodeForces => 800 rating

link: https://codeforces.com/problemset/problem/2030/B

B. Minimise Oneness

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes

For an arbitrary binary string t^*, let $f(t)$ be the number of non-empty subsequences† of t that contain only 0, and let $g(t)$ be the number of non-empty subsequences of t that contain at least one 1.

Note that for $f(t)$ and for $g(t)$, each subsequence is counted as many times as it appears in t. E.g., $f(000) = 7, g(100) = 4$.

We define the *oneness* of the binary string t to be $|f(t) - g(t)|$, where for an arbitrary integer z, $|z|$ represents the absolute value of z.

You are given a positive integer n. Find a binary string s of length n such that its *oneness* is as small as possible. If there are multiple strings, you can print any of them.

*A binary string is a string that only consists of characters 0 and 1.

†A sequence a is a subsequence of a sequence b if a can be obtained from b by the deletion of several (possibly, zero or all) elements. For example, subsequences of 1011101 are 0, 1, 11111, 0111, but not 000 nor 11100.

Input

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

The only line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the length of s.

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

Output

For each test case, output s on a new line. If multiple answers exist, output any.

Example

input Copy

```
3
1
2
3
```

output Copy

```
0
01
010
```

Note

In the first test case, for the example output, $f(t) = 1$ because there is one subsequence that contains only 0 (0), and $g(t) = 0$ because there are no subsequences that contain at least one 1. The *oneness* is $|1 - 0| = 1$. The output 1 is correct as well because its *oneness* is $|0 - 1| = 1$.

For the example output of the second test case, $f(t) = 1$ because there is one non-empty subsequence that contains only 0, and $g(t) = 2$ because there are two non-empty subsequences that contain at least one 1 (01 and 1). The *oneness* is thus $|1 - 2| = 1$. It can be shown that 1 is the minimum possible value of its *oneness* over all possible binary strings of size 2.

QUESTION 2: Two Screens => CodeForces => 800 rating

link: https://codeforces.com/problemset/problem/2025/A

There are two screens which can display sequences of uppercase Latin letters. Initially, both screens display nothing.

In one second, you can do one of the following two actions:

- choose a screen and an uppercase Latin letter, and append that letter to **the end** of the sequence displayed on that screen;
- choose a screen and copy the sequence from it to the other screen, **overwriting the sequence that was displayed on the other screen**.

You have to calculate the minimum number of seconds you have to spend so that the first screen displays the sequence s, and the second screen displays the sequence t.

Input

The first line contains one integer q ($1 \le q \le 500$) — the number of test cases.

Each test case consists of two lines. The first line contains the string s, and the second line contains the string t ($1 \le |s|, |t| \le 100$). Both strings consist of uppercase Latin letters.

Output

For each test case, print one integer — the minimum possible number of seconds you have to spend so that the first screen displays the sequence s, and the second screen displays the sequence t.

Example

input Copy

```
3
GARAGE
GARAGEFORSALE
ABCDE
AABCD
TRAINING
DRAINING
```

output Copy

```
14
10
16
```

Note

In the first test case, the following sequence of actions is possible:

- spend 6 seconds to write the sequence GARAGE on the first screen;
- copy the sequence from the first screen to the second screen;
- spend 7 seconds to complete the sequence on the second screen by writing FORSALE.

In the second test case, the following sequence of actions is possible:

- spend 1 second to write the sequence A on the second screen;
- copy the sequence from the second screen to the first screen;
- spend 4 seconds to complete the sequence on the first screen by writing BCDE;
- spend 4 seconds to complete the sequence on the second screen by writing ABCD.

In the third test case, the fastest way to display the sequences is to type both of them character by character without copying, and this requires 16 seconds.

QUESTION 3: Bus to Pénjamo => CodeForces => 800 rating

link: https://codeforces.com/problemset/problem/2022/A

There are n families travelling to Pénjamo to witness Mexico's largest-ever "walking a chicken on a leash" marathon. The i-th family has a_i family members. All families will travel using a single bus consisting of r rows with 2 seats each.

A person is considered *happy* if:

- Another family member is seated in the same row as them, or
- They are sitting alone in their row (with an empty seat next to them).

Determine the maximum number of happy people in an optimal seating arrangement. Note that **everyone** must be seated in the bus.

It is guaranteed that all family members will fit on the bus. Formally, it is guaranteed that $\sum_{i=1}^{n} a_i \leq 2r$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 1000$). The description of the test cases follows.

The first line of each test case contains two integers n and r ($1 \leq n \leq 100$; $1 \leq r \leq 500$) — the number of families and the number of rows in the bus.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10$) — the number of family members in each family.

Output

For each test case, output the maximum number of happy people in an optimal seating arrangement.

Example

input Copy

```
4
3 3
2 3 1
3 3
2 2 2
4 5
1 1 2 2
4 5
3 1 1 3
```

output Copy

```
4
6
6
6
```

Note

In the first test case, the two members of the first family can sit together in the first row, while the two members of the second family can sit together in the second row. The remaining member of the second family can sit in the third row along with a member of the third family. This seating arrangement is shown below, where the 4 happy people are colored green.

1	1
2	2
2	3

In the second test case, a possible seating arrangement with 6 happy people is shown below.

3	3
1	1
2	2

In the third test case, a possible seating arrangement with 6 happy people is shown below.

4	4
	2
3	3
1	