Codeforces Round #634 (Div. 3)

A. Candies and Two Sisters

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

There are two sisters Alice and Betty. You have n candies. You want to distribute these n candies between two sisters in such a way that:

- Alice will get a ($a > 0$) candies;
- Betty will get b ($b > 0$) candies;
- each sister will get some **integer** number of candies;
- Alice will get a greater amount of candies than Betty (i.e. $a > b$);
- all the candies will be given to one of two sisters (i.e. $a + b = n$).

Your task is to calculate the number of ways to distribute exactly n candies between sisters in a way described above. Candies are indistinguishable.

Formally, find the number of ways to represent n as the sum of $n = a + b$, where a and b are positive integers and $a > b$.

You have to answer t independent test cases.

Input

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

The only line of a test case contains one integer n ($1 \le n \le 2 \cdot 10^9$) — the number of candies you have.

Output

For each test case, print the answer — the number of ways to distribute exactly n candies between two sisters in a way described in the problem statement. If there is no way to satisfy all the conditions, print 0.

Example

input
```
6
7
1
2
3
2000000000
763243547
```
output
```
3
0
0
1
999999999
381621773
```

Note

For the test case of the example, the 3 possible ways to distribute candies are:

- $a = 6$, $b = 1$;
- $a = 5$, $b = 2$;
- $a = 4$, $b = 3$.

B. Construct the String

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

You are given three positive integers n, a and b. You have to construct a string s of length n consisting of lowercase Latin letters such that **each substring** of length a has **exactly** b distinct letters. It is guaranteed that the answer exists.

You have to answer t independent test cases.

Recall that the substring $s[l \dots r]$ is the string $s_l, s_{l+1}, \dots, s_r$ and its length is $r - l + 1$. In this problem you are only interested in substrings of length a.

Input

The first line of the input contains one integer t ($1 \le t \le 2000$) — the number of test cases. Then t test cases follow.

The only line of a test case contains three space-separated integers n, a and b ($1 \le a \le n \le 2000, 1 \le b \le \min(26, a)$), where n is the length of the required string, a is the length of a substring and b is the required number of distinct letters in each substring of length a.

It is guaranteed that the sum of n over all test cases does not exceed 2000 ($\sum n \le 2000$).

Output

For each test case, print the answer — such a string s of length n consisting of lowercase Latin letters that **each substring** of length a has **exactly** b distinct letters. If there are multiple valid answers, print any of them. It is guaranteed that the answer exists.

Example

input

```
4
7 5 3
6 1 1
6 6 1
5 2 2
```

output

```
tleelte
qwerty
vvvvvv
abcde
```

Note

In the first test case of the example, consider all the substrings of length 5:

- "`tleel`": it contains 3 distinct (unique) letters,
- "`leelt`": it contains 3 distinct (unique) letters,
- "`eelte`": it contains 3 distinct (unique) letters.

C. Two Teams Composing

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

You have n students under your control and you have to compose **exactly two teams** consisting of some subset of your students. Each student had his own skill, the i-th student skill is denoted by an integer a_i (different students can have the same skills).

So, about the teams. Firstly, these two teams should have the same size. Two more constraints:

- The first team should consist of students with **distinct** skills (i.e. all skills in the first team are unique).
- The second team should consist of students with **the same** skills (i.e. all skills in the second team are equal).

Note that it is permissible that some student of the first team has the same skill as a student of the second team.

Consider some examples (skills are given):

- $[1, 2, 3]$, $[4, 4]$ is not a good pair of teams because sizes should be the same;
- $[1, 1, 2]$, $[3, 3, 3]$ is not a good pair of teams because the first team should not contain students with the same skills;
- $[1, 2, 3]$, $[3, 4, 4]$ is not a good pair of teams because the second team should contain students with the same skills;
- $[1, 2, 3]$, $[3, 3, 3]$ is a good pair of teams;
- $[5]$, $[6]$ is a good pair of teams.

Your task is to find the maximum possible size x for which it is possible to compose a valid pair of teams, where each team size is x (skills in the first team needed to be unique, skills in the second team should be the same between them). A student cannot be part of more than one team.

You have to answer t independent test cases.

Input

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

The first line of the test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of students. The second line of the test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$), where a_i is the skill of the i-th student. Different students can have the same skills.

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

Output

For each test case, print the answer — the maximum possible size x for which it is possible to compose a valid pair of teams, where each team size is x.

Example

input

```
4
7
4 2 4 1 4 3 4
5
2 1 5 4 3
1
1
4
1 1 1 3
```

output

```
3
1
0
2
```

Note

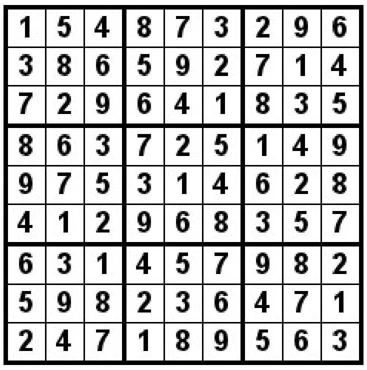
In the first test case of the example, it is possible to construct two teams of size 3: the first team is $[1, 2, 4]$ and the second team is $[4, 4, 4]$. Note, that there are some other ways to construct two valid teams of size 3.

D. Anti-Sudoku

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

You are given a correct solution of the sudoku puzzle. If you don't know what is the sudoku, you can read about it here.

The picture showing the correct sudoku solution:



1	5	4	8	7	3	2	9	6
3	8	6	5	9	2	7	1	4
7	2	9	6	4	1	8	3	5
8	6	3	7	2	5	1	4	9
9	7	5	3	1	4	6	2	8
4	1	2	9	6	8	3	5	7
6	3	1	4	5	7	9	8	2
5	9	8	2	3	6	4	7	1
2	4	7	1	8	9	5	6	3

Blocks are bordered with bold black color.

Your task is to change **at most** 9 elements of this field (i.e. choose some $1 \le i, j \le 9$ and change the number at the position (i, j) to any other number in range $[1; 9]$) to make it **anti-sudoku**. The **anti-sudoku** is the 9×9 field, in which:

- Any number in this field is in range $[1; 9]$;
- each row contains at least two equal elements;
- each column contains at least two equal elements;
- each 3×3 block (you can read what is the block in the link above) contains at least two equal elements.

It is guaranteed that the answer exists.

You have to answer t independent test cases.

Input

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

Each test case consists of 9 lines, each line consists of 9 characters from 1 to 9 without any whitespaces — the correct solution of the sudoku puzzle.

Output

For each test case, print the answer — the initial field with **at most** 9 changed elements so that the obtained field is **anti-sudoku**. If there are several solutions, you can print any. It is guaranteed that the answer exists.

Example

input

```
1
154873296
386592714
729641835
863725149
975314628
```

```
412968357
631457982
598236471
247189563
```

output

```
154873396
336592714
729645835
863725145
979314628
412958357
631457992
998236471
247789563
```

E1. Three Blocks Palindrome (easy version)

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between easy and hard versions is constraints.

You are given a sequence a consisting of n positive integers.

Let's define a **three blocks palindrome** as the sequence, consisting of *at most two* distinct elements (let these elements are a and b, a can be equal b) and is as follows: $[\underbrace{a, a, \dots, a}_{x}, \underbrace{b, b, \dots, b}_{y}, \underbrace{a, a, \dots, a}_{x}]$. There x, y are integers greater than or equal to 0. For example, sequences $[]$, $[2]$, $[1, 1]$, $[1, 2, 1]$, $[1, 2, 2, 1]$ and $[1, 1, 2, 1, 1]$ are **three block palindromes** but $[1, 2, 3, 2, 1]$, $[1, 2, 1, 2, 1]$ and $[1, 2]$ are not.

Your task is to choose the maximum by length **subsequence** of a that is a **three blocks palindrome**.

You have to answer t independent test cases.

Recall that the sequence t is a a subsequence of the sequence s if t can be derived from s by removing zero or more elements without changing the order of the remaining elements. For example, if $s = [1, 2, 1, 3, 1, 2, 1]$, then possible subsequences are: $[1, 1, 1, 1]$, $[3]$ and $[1, 2, 1, 3, 1, 2, 1]$, but not $[3, 2, 3]$ and $[1, 1, 1, 1, 2]$.

Input

The first line of the input contains one integer t ($1 \le t \le 2000$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($1 \le n \le 2000$) — the length of a. The second line of the test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 26$), where a_i is the i-th element of a. **Note that the maximum value of a_i can be up to 26**.

It is guaranteed that the sum of n over all test cases does not exceed 2000 ($\sum n \le 2000$).

Output

For each test case, print the answer — the maximum possible length of some subsequence of a that is a **three blocks palindrome**.

Example

input

```
6
8
1 1 2 2 3 2 1 1
3
1 3 3
4
1 10 10 1
1
26
2
2 1
3
1 1 1
```

output

```
7
2
4
1
1
3
```

E2. Three Blocks Palindrome (hard version)

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

output: standard output

The only difference between easy and hard versions is constraints.

You are given a sequence a consisting of n positive integers.

Let's define a **three blocks palindrome** as the sequence, consisting of *at most two* distinct elements (let these elements are a and b, a can be equal b) and is as follows: $[\underbrace{a, a, \dots, a}_{x}, \underbrace{b, b, \dots, b}_{y}, \underbrace{a, a, \dots, a}_{x}]$. There x, y are integers greater than or equal to 0. For example, sequences $[]$, $[2]$, $[1, 1]$, $[1, 2, 1]$, $[1, 2, 2, 1]$ and $[1, 1, 2, 1, 1]$ are **three block palindromes** but $[1, 2, 3, 2, 1]$, $[1, 2, 1, 2, 1]$ and $[1, 2]$ are not.

Your task is to choose the maximum by length **subsequence** of a that is a **three blocks palindrome**.

You have to answer t independent test cases.

Recall that the sequence t is a a subsequence of the sequence s if t can be derived from s by removing zero or more elements without changing the order of the remaining elements. For example, if $s = [1, 2, 1, 3, 1, 2, 1]$, then possible subsequences are: $[1, 1, 1, 1]$, $[3]$ and $[1, 2, 1, 3, 1, 2, 1]$, but not $[3, 2, 3]$ and $[1, 1, 1, 1, 2]$.

Input

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

The first line of the test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the length of a. The second line of the test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 200$), where a_i is the i-th element of a. **Note that the maximum value of a_i can be up to 200**.

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

Output

For each test case, print the answer — the maximum possible length of some subsequence of a that is a **three blocks palindrome**.

Example

input

```
6
8
1 1 2 2 3 2 1 1
3
1 3 3
4
1 10 10 1
1
26
2
2 1
3
1 1 1
```

output

```
7
2
4
1
1
3
```

F. Robots on a Grid

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is a rectangular grid of size $n \times m$. Each cell of the grid is colored black ('0') or white ('1'). The color of the cell (i, j) is $c_{i,j}$. You are also given a map of directions: for each cell, there is a direction $s_{i,j}$ which is one of the four characters 'U', 'R', 'D' and 'L'.

- If $s_{i,j}$ is 'U' then there is a transition from the cell (i, j) to the cell $(i - 1, j)$;
- if $s_{i,j}$ is 'R' then there is a transition from the cell (i, j) to the cell $(i, j + 1)$;
- if $s_{i,j}$ is 'D' then there is a transition from the cell (i, j) to the cell $(i + 1, j)$;
- if $s_{i,j}$ is 'L' then there is a transition from the cell (i, j) to the cell $(i, j - 1)$.

It is guaranteed that the top row doesn't contain characters 'U', the bottom row doesn't contain characters 'D', the leftmost column doesn't contain characters 'L' and the rightmost column doesn't contain characters 'R'.

You want to place some robots in this field (at most one robot in a cell). The following conditions should be satisfied.

- Firstly, each robot **should move** every time (i.e. it cannot skip the move). **During one move each robot goes to the adjacent cell depending on the current direction**.

- Secondly, you have to place robots in such a way that **there is no move** before which two different robots occupy the same cell (it also means that you cannot place two robots in the same cell). I.e. if the grid is "RL" (one row, two columns, colors does not matter there) then you can place two robots in cells $(1, 1)$ and $(1, 2)$, but if the grid is "RLL" then you cannot place robots in cells $(1, 1)$ and $(1, 3)$ because during the first second both robots will occupy the cell $(1, 2)$.

The robots make an infinite number of moves.

Your task is to place the maximum number of robots to satisfy all the conditions described above and among all such ways, you have to choose one where the number of **black** cells occupied by robots **before all movements** is the maximum possible. **Note that you can place robots only before all movements**.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 5 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The first line of the test case contains two integers n and m ($1 < nm \le 10^6$) — the number of rows and the number of columns correspondingly.

The next n lines contain m characters each, where the j-th character of the i-th line is $c_{i,j}$ ($c_{i,j}$ is either '0' if the cell (i, j) is black or '1' if the cell (i, j) is white).

The next n lines also contain m characters each, where the j-th character of the i-th line is $s_{i,j}$ ($s_{i,j}$ is 'U', 'R', 'D' or 'L' and describes the direction of the cell (i, j)).

It is guaranteed that the sum of the sizes of fields does not exceed 10^6 ($\sum nm \le 10^6$).

Output

For each test case, print two integers — the maximum number of robots you can place to satisfy all the conditions described in the problem statement and the maximum number of **black** cells occupied by robots **before all movements** if the number of robots placed is maximized. **Note that you can place robots only before all movements**.

Example

input

```
3
1 2
01
RL
3 3
001
101
110
RLL
DLD
ULL
3 3
000
000
000
RRD
RLD
ULL
```

output

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