Codeforces Round 952 (Div. 4)

A. Creating Words

1 second, 256 megabytes

Matthew is given two strings a and b, both of length 3. He thinks it's particularly funny to create two new words by swapping the first character of a with the first character of b. He wants you to output a and b after the swap.

Note that the new words may not necessarily be different.

Input

The first line contains t $(1 \le t \le 100)$ — the number of test cases.

The first and only line of each test case contains two space-separated strings, a and b, both of length 3. The strings only contain lowercase Latin letters.

Output

For each test case, after the swap, output a and b, separated by a space.

input
```
6
bit set
cat dog
hot dog
uwu owo
cat cat
zzz zzz
```

output
```
sit bet
dat cog
dot hog
owu uwo
cat cat
zzz zzz
```

B. Maximum Multiple Sum

1 second, 256 megabytes

Given an integer n, find an integer x such that:

- $2 \le x \le n$.
- The sum of multiples of x that are less than or equal to n is maximized. Formally, $x + 2x + 3x + \dots + kx$ where $kx \le n$ is maximized over all possible values of x.

Input

The first line contains t $(1 \le t \le 100)$ — the number of test cases.

Each test case contains a single integer n $(2 \le n \le 100)$.

Output

For each test case, output an integer, the optimal value of x. It can be shown there is only one unique answer.

input
```
2
3
15
```

output
```
3
2
```

For $n = 3$, the possible values of x are 2 and 3. The sum of all multiples of 2 less than or equal to n is just 2, and the sum of all multiples of 3 less than or equal to n is 3. Therefore, 3 is the optimal value of x.

For $n = 15$, the optimal value of x is 2. The sum of all multiples of 2 less than or equal to n is $2 + 4 + 6 + 8 + 10 + 12 + 14 = 56$, which can be proven to be the maximal over all other possible values of x.

C. Good Prefixes

2 seconds, 256 megabytes

Alex thinks some array is *good* if there exists some element that can be represented as the sum of all **other** elements (the sum of all other elements is 0 if there are no other elements). For example, the array $[1, 6, 3, 2]$ is good since $1 + 3 + 2 = 6$. Furthermore, the array $[0]$ is also good. However, the arrays $[1, 2, 3, 4]$ and $[1]$ are not good.

Alex has an array $a_1, a_2, \dots, a_n$. Help him count the number of good non-empty prefixes of the array a. In other words, count the number of integers i $(1 \le i \le n)$ such that the length i prefix (i.e. $a_1, a_2, \dots, a_i$) is good.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of elements in the array.

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

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 a single integer — the number of good non-empty prefixes of the array a.

input
```
7
1
0
1
1
4
1 1 2 0
5
0 1 2 1 4
7
1 1 0 3 5 2 12
7
1000000000 1000000000 1000000000 1000000000 1000000000 1000000000
294967296
10
0 1000000000 1000000000 1000000000 1000000000 1000000000
1000000000 1000000000 1000000000 589934592
```

output
```
1
0
3
3
4
1
2
```

In the fourth test case, the array has five prefixes:

- prefix $[0]$ is a good array, as mentioned in the statement;
- prefix $[0, 1]$ is not a good array, since $0 \ne 1$;
- prefix $[0, 1, 2]$ is not a good array, since $0 \ne 1 + 2$, $1 \ne 0 + 2$ and $2 \ne 0 + 1$;
- prefix $[0, 1, 2, 1]$ is a good array, since $2 = 0 + 1 + 1$;
- prefix $[0, 1, 2, 1, 4]$ is a good array, since $4 = 0 + 1 + 2 + 1$.

As you can see, three of them are good, so the answer is 3.

D. Manhattan Circle

2 seconds, 256 megabytes

Given a n by m grid consisting of '.' and '#' characters, there exists a whole manhattan circle on the grid. The top left corner of the grid has coordinates $(1,1)$, and the bottom right corner has coordinates (n,m).

Point (a,b) belongs to the manhattan circle centered at (h,k) if $|h-a|+|k-b|<r$, where r is a positive constant.

On the grid, the set of points that are part of the manhattan circle is marked as '#'. Find the coordinates of the center of the circle.

Input

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

The first line of each test case contains n and m ($1 \le n \cdot m \le 2 \cdot 10^5$) — the height and width of the grid, respectively.

The next n lines contains m characters '.' or '#'. If the character is '#', then the point is part of the manhattan circle.

It is guaranteed the sum of $n \cdot m$ over all test cases does not exceed $2 \cdot 10^5$, and there is a whole manhattan circle on the grid.

Output

For each test case, output the two integers, the coordinates of the center of the circle.

input

```
6
5 5
.....
.....
..#..
.....
.....
5 5
..#..
.###.
#####
.###.
..#..
5 6
......
......
.#....
###...
.#....
1 1
#
5 6
...#..
..###.
.#####
..###.
...#..
2 10
..........
...#......
```

output

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

E. Secret Box

1 second, 256 megabytes

Ntarsis has a box B with side lengths x, y, and z. It lies in the 3D coordinate plane, extending from $(0,0,0)$ to (x,y,z).

Ntarsis has a secret box S. He wants to choose its dimensions such that all side lengths are positive integers, and the volume of S is k. He can place S somewhere within B such that:

- S is parallel to all axes.
- every corner of S lies on an integer coordinate.

S is magical, so when placed at an integer location inside B, it will not fall to the ground.

Among all possible ways to choose the dimensions of S, determine the **maximum** number of distinct locations he can choose to place his secret box S inside B. Ntarsis does not rotate S once its side lengths are selected.

Input

The first line consists of an integer t, the number of test cases ($1 \le t \le 2000$). The description of the test cases follows.

The first and only line of each test case contains four integers x, y, z and k ($1 \le x, y, z \le 2000$, $1 \le k \le x \cdot y \cdot z$).

It is guaranteed the sum of all x, sum of all y, and sum of all z do not exceed 2000 over all test cases.

Note that k may not fit in a standard 32-bit integer data type.

Output

For each test case, output the answer as an integer on a new line. If there is no way to select the dimensions of S so it fits in B, output 0.

input

```
7
3 3 3 8
3 3 3 18
5 1 1 1
2 2 2 7
3 4 2 12
4 3 1 6
1800 1800 1800 4913000000
```

output

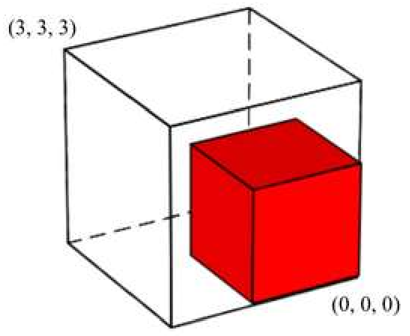
```
8
2
5
0
4
4
1030301
```

For the first test case, it is optimal to choose S with side lengths 2, 2, and 2, which has a volume of $2 \cdot 2 \cdot 2 = 8$. It can be shown there are 8 ways to put S inside B.

The coordinate with the least x, y, and z values for each possible arrangement of S are:

1. $(0,0,0)$
2. $(1,0,0)$
3. $(0,1,0)$
4. $(0,0,1)$
5. $(1,0,1)$
6. $(1,1,0)$
7. $(0,1,1)$
8. $(1,1,1)$

The arrangement of S with a coordinate of $(0,0,0)$ is depicted below:



For the second test case, S with side lengths 2, 3, and 3 are optimal.

F. Final Boss

2 seconds, 256 megabytes

You are facing the final boss in your favorite video game. The boss enemy has h health. Your character has n attacks. The i'th attack deals a_i damage to the boss but has a cooldown of c_i turns, meaning the next time you can use this attack is turn $x + c_i$ if your current turn is x. Each turn, you can use all attacks that are not currently on cooldown, **all at once**. If all attacks are on cooldown, you do nothing for the turn and skip to the next turn.

Initially, all attacks are not on cooldown. How many turns will you take to beat the boss? The boss is beaten when its health is 0 or less.

Input

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

The first line of each test case contains two integers h and n ($1 \le h, n \le 2 \cdot 10^5$) – the health of the boss and the number of attacks you have.

The following line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 2 \cdot 10^5$) – the damage of your attacks.

The following line of each test case contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 2 \cdot 10^5$) – the cooldown of your attacks.

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

Output

For each test case, output an integer, the minimum number of turns required to beat the boss.

input

```
8
3 2
2 1
2 1
5 2
2 1
2 1
50 3
5 6 7
5 6 7
50 3
2 2 2
3 3 3
90000 2
200000 200000
1 1
100000 1
1
200000
6 7
3 2 3 2 3 1 2
6 5 9 5 10 7 7
21 6
1 1 1 1 1 1
5 5 8 10 7 6
```

output

```
1
3
15
25
1
19999800001
1
21
```

For the first test case, you can use attacks 1 and 2 on the first turn, dealing 3 damage in total, and slaying the boss.

For the second case, you can beat the boss in 3 turns by using the following attacks:

Turn 1: Use attacks 1 and 2, dealing 3 damage to the boss. The boss now has 2 health left.

Turn 2: Use attack 2, dealing 1 damage to the boss. The boss now has 1 health left.

Turn 3: Use attack 1, dealing 2 damage to the boss. The boss now has -1 health left. Since its health is less than or equal to 0, you beat the boss.

For the sixth test case: remember to use 64-bit integers as the answer can get large.

G. D-Function

2 seconds, 256 megabytes

Let $D(n)$ represent the sum of digits of n. For how many integers n where $10^l \le n < 10^r$ satisfy $D(k \cdot n) = k \cdot D(n)$? Output the answer modulo $10^9 + 7$.

Input

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

Each test case contains three integers l, r, and k ($0 \le l < r \le 10^9$, $1 \le k \le 10^9$).

Output

For each test case, output an integer, the answer, modulo $10^9 + 7$.

input

```
6
0 1 4
0 2 7
1 2 1
1 2 3
582 74663 3
0 3 1
```

output

```
2
3
90
12
974995667
999
```

For the first test case, the only values of n that satisfy the condition are 1 and 2.

For the second test case, the only values of n that satisfy the condition are 1, 10, and 11.

For the third test case, all values of n between 10 inclusive and 100 exclusive satisfy the condition.

H1. Maximize the Largest Component (Easy Version)

2 seconds, 512 megabytes

Easy and hard versions are actually different problems, so read statements of both problems completely and carefully. The only difference between the two versions is the operation.

Alex has a grid with n rows and m columns consisting of '.' and '#' characters. A set of '#' cells forms a connected component if from any cell in this set, it is possible to reach any other cell in this set by only moving to another cell in the set that shares a **common side**. The size of a connected component is the number of cells in the set.

In one operation, Alex selects any row r $(1 \le r \le n)$ **or** any column c ($1 \le c \le m$), then sets every cell in row r **or** column c to be '#'. Help Alex find the maximum possible size of the largest connected component of '#' cells that he can achieve after performing the operation **at most once**.

Input

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

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

The next n lines each contain m characters. Each character is either '.' or '#'.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed 10^6.

Output

For each test case, output a single integer — the maximum possible size of a connected component of '#' cells that Alex can achieve.

input

```
6
1 1
.
4 2
..
#.
#.
.#
3 5
.#.#.
..#..
.#.#.
5 5
#...#
....#
#...#
.....
...##
6 6
.#..#.
#..#..
.#...#
#.#.#.
.#.##.
###..#
6 8
..#....#
.####.#.
###.#..#
.##.#.##
.#.##.##
#..##.#.
```

output

```
1
6
9
11
15
30
```

In the second test case, it is optimal for Alex to set all cells in column 2 to be '#'. Doing so will lead to the largest connected component of '#' having a size of 6.

In the third test case, it is optimal for Alex to set all cells in row 2 to be '#'. Doing so will lead to the largest connected component of '#' having a size of 9.

In the fourth test case, it is optimal for Alex to set all cells in row 4 to be '#'. Doing so will lead to the largest connected component of '#' having a size of 11.

H2. Maximize the Largest Component (Hard Version)

2 seconds, 512 megabytes

Easy and hard versions are actually different problems, so read statements of both problems completely and carefully. The only difference between the two versions is the operation.

Alex has a grid with n rows and m columns consisting of '.' and '#' characters. A set of '#' cells forms a connected component if from any cell in this set, it is possible to reach any other cell in this set by only moving to another cell in the set that shares a **common side**. The size of a connected component is the number of cells in the set.

In one operation, Alex selects any row r ($1 \le r \le n$) **and** any column c ($1 \le c \le m$), then sets every cell in row r **and** column c to be '#'. Help Alex find the maximum possible size of the largest connected component of '#' cells that he can achieve after performing the operation **at most once**.

Input

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

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

The next n lines each contain m characters. Each character is either '.' or '#'.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed 10^6.

Output

For each test case, output a single integer — the maximum possible size of a connected component of '#' cells that Alex can achieve.

input

```
6
1 1
.
4 2
..
#.
#.
.#
3 5
.#.#.
..#..
.#.#.
5 5
#...#
....#
#...#
.....
...##
6 6
.#..#.
#..#..
.#...#
#.#.#.
.#.##.
###..#
6 8
..#....#
.####.#.
###.#..#
.##.#.##
.#.##.##
#..##.#.
```

output

```
1
7
11
16
22
36
```

In the fourth test case, it is optimal for Alex to set all cells in row 4 and column 2 to be '#'. Doing so will lead to the largest connected component of '#' having a size of 16.

In the fifth test case, it is optimal for Alex to set all cells in row 2 and column 4 to be '#'. Doing so will lead to the largest connected component of '#' having a size of 22.