Greetings From Globussoft

- Given below are 5 Programming questions, you have to solve any 3 out of 5 questions.

- These 5 questions you can attempt in any technology like C/C++, java, .Net, PHP

- To solve these 3 questions you've max. 3 hours.

- While Solving these questions you are not allowed to use any **Search Engine** like Google, Yahoo, Bing …

All the best for your test

Globussoft

QUESTION - 1

Shortly after his abdication from the Bytelandian throne Johnny decided to go into farming. Water melons were a natural choice as his first crop ever, since they seemed easy enough to grow and look after. So, he sold all his beer bottles and for the money he purchased a 1km x 1km square field. Here it was that he planted the water melon seeds. (The word 'planted' is really a bit of a euphemism for walking across a field gorging on a water melon and spitting out the pips but, for the sake of politeness, let us leave it this way).

To everyone's surprise a lot of the seeds sprouted stems, and soon enough many of the plants showed signs of fruit (and some had even more than one!). Then quite unexpectedly, when the water melons were still a little too unripe to eat, winter set in. Johnny knows that he has to construct a green house to protect the field but, with his rather limited budget, he cannot afford the glass to cover the whole area. He has decided that it is enough that *k* fruit survive the ordeal under a glazed roof. For reasons of architectural planning in Byteland it is necessary that the green house be a rectangle with sides parallel to the edges of the plot.

You have been requested to help Johnny minimise investement costs. Since glass is paid for by the square meter, design a green house with the smallest possible area fulfilling the imposed conditions.

Input

The first line of input contains the integer $t<=100$, the number of test cases. *t* test cases follow.

Every test case begins with a line containing two integers *n k*, denoting the total number of plants and the number of water melon fruit to be protected, respectively ($1<=n<=1000$, $1<=k<=10^6$, *k* doesn't exceed the total number of fruit in the plantation). Each of the next *n* lines describes a single plant, the *i*-th line containing three integers $x_i\ y_i\ f_i$ - the X and Y coordinates of the plant, and the number of water melon fruit on it, respectively ($1<=x_i, y_i, f_i<=1000$).

Output

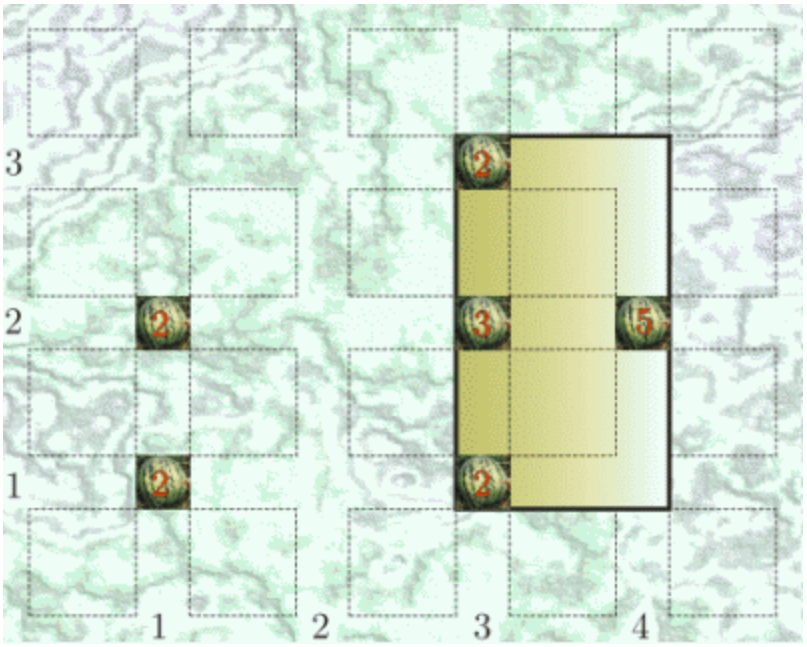
For each test case output a single integer, denoting the area of the smallest possible rectangular glass house with horizontal and vertical edges, sufficient to cover at least *k* fruit of the plantation.

Example

Input:

```
1
6 11
1 1 2
1 2 2
3 1 2
3 2 3
4 2 5
3 3 2
```

Output:
2



QUESTION – 2

There is a land far, far away were the entire population dwells in walled cities at the peaks of mountains on the circumference of a plateau known as The Circle. The High Councillors of the cities developed an intricate system of communication: the cities were connected into a cycle by a perfectly round waterway. If need arose, a small paper boat with a message tied to its sail was released into the waterway and was guided by its solitary crew member (a small tin soldier) from one city to the next, and so on, until it reached its destination. Some segments of the waterway were only passable in one direction (due to waterfalls), and so there may have been pairs of cities for which communication was impossible.

As the centuries went by, the system slowly began to show its weaknesses. The waterway was so narrow that two boats going in opposite directions could never pass each other. To make matters worse, some of the more enterprising cities replaced the tin soldier by a plastic one to increase the speed of the boat, and the faster boats had to queue up behind the slower ones, and everyone got very angry indeed. The councillors gathered to address the problem and found that the best course of action would be to construct two separate channels between every pair of communicating cities A and B: one for carrying messages from A to B, the other from B to A (if communication was impossible in some direction in the old waterway, it needn't be enabled in the new one).

The High Priests of the Circle were the first to protest against the plan. They insisted that any waterway ever built should be circular and go round all the cities in the same manner as the original one, and the route of any boat must always be a perfect arc between any two adjacent cities. So the newly designed channels would in fact have to be composed of sets of adjacent fragments of circles, without any two channels sharing an arc.

The engineers have quite rightly pointed out that the new circles will be prone to the same problem of waterfalls on the same sections as the original waterway. Bearing this in mind, given a map of the old waterway, calculate the smallest possible number of circles the new waterway may consist of.

Input

Input begins with integer t<=100, the number of test cases. t test cases follow.

Each test case consists of two lines. The first contains a single integer n (3<=n<=100000), the number of cities around the Circle. The second line is a description of the old waterway - a sequence of exactly n characters 'A', 'B' or 'C', without separating spaces, terminated by a new line. These characters correspond to the state of the arcs between cities 1 and 2, 2 and 3,..., n-1 and n, n and 1, respectively, and mean: 'A' - the arc is passable when going anticlockwise, 'B' - the arc is passable in both directions, 'C' - the arc is passable when going clockwise.

Output

For each test case output a line, containing a single integer - the number of circles required for the new waterway.

Example

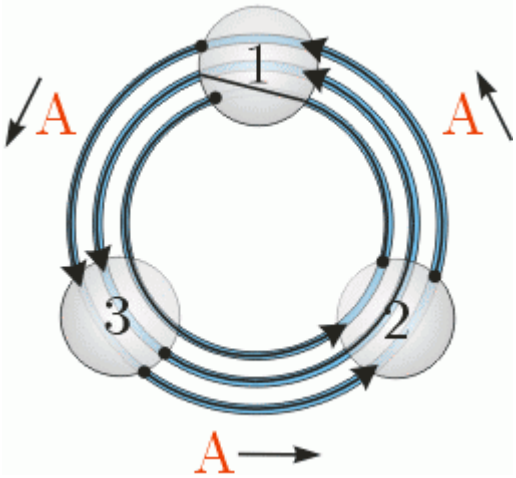
Input:

```
2
3
AAA
4
BACB
```

Output:

```
3
5
```

A solution to the first test case which requires 3 circles is presented below.



QUESTION – 3

As you are probably well aware, in Byteland it is always the military officer's main worry to order his soldiers on parade correctly. In Bitland ordering soldiers is not really such a problem. If a platoon consists of n men, all of them have different rank (from 1 - lowest to n - highest) and on parade they should be lined up from left to right in increasing order of rank.

Sounds simple, doesn't it? Well, Msgt Johnny thought the same, until one day he was faced with a new command. He soon discovered that his elite commandos preferred to do the fighting, and leave the thinking to their superiors. So, when at the first rollcall the soldiers lined up in fairly random order it was not because of their lack of discipline, but simply because they couldn't work out how to form a line in correct order of ranks. Msgt Johnny was not at all amused, particularly as he soon found that none of the soldiers even remembered his own rank. Over the years of service every soldier had only learned which of the other soldiers were his superiors. But Msgt Johnny was not a man to give up easily when faced with a true military challenge. After a moment's thought a solution of brilliant simplicity struck him and he issued the following order: `"men, starting from the left, one by one, do: (step forward; go left until there is no superior to the left of you; get back in line)."`. This did indeed get the men sorted in a few minutes. The problem was solved... for the time being.

The next day, the soldiers came in exactly the same order as the day before, and had to be rearranged using the same method. History repeated. After some weeks, Msgt Johnny managed to force each of his soldiers to remember how many men he passed when going left, and thus make the sorting process even faster.

If you know how many positions each man has to walk to the left, can you try to find out what order of ranks the soldiers initially line up in?

Input

The first line of input contains an integer t<=50, the number of test cases. It is followed by t test cases, each consisting of 2 lines. The first line contains a single integer n (1<=n<=200000). The second line contains n space separated integers w_i, denoting how far the i-th soldier in line must walk to the left when applying Msgt Johnny's algorithm.

Output

For each test case, output a single line consisting of n space separated integers - the ranks of the soldiers, given from left to right in their initial arrangement.

Example

Input:

```
2
3
0 1 0
5
0 1 2 0 1
```

Output:

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

QUESTION – 4

The floor of a store is a rectangle divided into *n*m* square fields. Two fields are adjacent, if they have a common side. A parcel lays on one of the fields. Each of the remaining fields is either empty, or occupied by a case, which is too heavy to be moved by a store-keeper. The store-keeper has to shift the parcel from the starting field **P** to the final field **K**. He can move on the empty fields, going from the field on which he stands to a chosen adjacent field. When the store-keeper stays on a field adjacent to the one with the parcel he may push the parcel so that if moves to the next field (i.e. the field on the other side of the parcel), assuming condition that there are no cases on this field.

Task

Write a program, which:

- reads from the standard input a store scheme, a starting position of the store-keeper and a final position of the parcel,
- computes minimal number of the parcel shifts through borders of fields, which are necessary to put the parcel in the final position or decides that it is impossible to put the parcel there,
- writes the result into the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. In the first line of each test case two positive integers separated by a single space, *n,m<=100*, are written. These are dimensions of the store. In each of the following *n* lines there appears one *m*-letter word made of letters S, M, P, K, w. A letter on *i*-th position in *j*-th word denotes a type of the field with coordinates (*i,j*) and its meaning is following:

- S - case,
- M - starting position of the store-keeper,
- P - starting position of the parcel,
- K - final position of the parcel,
- w - empty field.

Each letter M, P and K appears in the test case exactly once.

Output

Your program should write to the standard output for each test case:

- exactly one word NO if the parcel cannot be put on the target field,
- exactly one integer, equal to the minimal number of the parcel shifts through borders of the fields, necessary to put a parcel on a final position, if it is possible to put the parcel there.

Example

Sample input:

```
1
10 12
SSSSSSSSSSS
Swwwwwww SSS
SwSSSSwwSSSS
SwSSSSwwSKSS
SwSSSSwwSwSS
SwwwwwPwwwww
SSSSSSSwSwSw
SSSSSSMwSwww
SSSSSSSSSSS
SSSSSSSSSSS
```

Sample output

```
7
```

QUESTION – 5

Rain has pummeled on the cows' field, a rectangular grid of R rows and C columns (1 <= R <= 50, 1 <= C <= 50). While good for the grass, the rain makes some patches of bare earth quite muddy. The cows, being meticulous grazers, don't want to get their hooves dirty while they eat.

To prevent those muddy hooves, Farmer John will place a number of wooden boards over the muddy parts of the cows' field. Each of the boards is 1 unit wide, and can be any length long. Each board must be aligned parallel to one of the sides of the field.

Farmer John wishes to minimize the number of boards needed to cover the muddy spots, some of which might require more than one board to cover. The boards may not cover any grass and deprive the cows of grazing area but they can overlap each other.

Compute the minimum number of boards FJ requires to cover all the mud in the field.

Input

t – the number of test cases, then *t* test cases follows.
Each test case is of the following form:
Two space-separated integers: *R* and *C*, then R lines follows
Each line contains a string of C characters, with '*' representing a muddy patch, and '.' representing a grassy patch. No spaces are present.

Output

For each test case output a single integer representing the number of boards FJ needs.

Example

Input:
```
1
4 4
*.*.
.***
***.
..*.
```

Output:
```
4
```

Output details:
```
Boards 1, 2, 3 and 4 are placed as follows:
1.2.
.333
444.
..2.
Board 2 overlaps boards 3 and 4.
```