Problem A Theme Park

Roller coasters are so much fun! It seems like everybody who visits the theme park wants to ride the roller coaster. Some people go alone; other people go in groups, and don't want to board the roller coaster unless they can all go together. And *everyone* who rides the roller coaster wants to ride again. A ride costs 1 Euro per person; your job is to figure out how much money the roller coaster will make today.

The roller coaster can hold k people at once. People queue for it in groups. Groups board the roller coaster, one at a time, until there are no more groups left or there is no room for the next group; then the roller coaster goes, whether it's full or not. Once the ride is over, all of its passengers re-queue in the same order. The roller coaster will run r times in a day.

For example, suppose $r = 4$, $k = 6$, and there are four groups of people with sizes: 1, 4, 2, 1. The first time the roller coaster goes, the first two groups $[1, 4]$ will ride, leaving an empty seat (the group of 2 won't fit, and the group of 1 can't go ahead of them). Then they'll go to the back of the queue, which now looks like 2, 1, 1, 4. The second time, the coaster will hold 4 people: $[2, 1, 1]$. Now the queue looks like 4, 2, 1, 1. The third time, it will hold 6 people: $[4, 2]$. Now the queue looks like $[1, 1, 4, 2]$. Finally, it will hold 6 people: $[1, 1, 4]$. The roller coaster has made a total of 21 Euros!

Input

The first line of the input contains three space-separated integers r, k and n ($1 \leq r \leq 1000$, $1 \leq k \leq 100$, $1 \leq n \leq 10$).

The second line contains n space-separated integers g_i ($1 \leq g_i \leq 10$), each of which is the size of a group that wants to ride. g_0 is the size of the first group, g_1 is the size of the second group, etc. It is guaranteed that all $g_i \leq k$.

Output

Output one line containing the number of Euros made by the roller coaster.

Examples

standard input	standard output
4 6 4 1 4 2 1	21
100 10 1 1	100
5 5 10 2 4 2 3 4 2 1 2 1 3	20

Problem B Snapper Chain

The *Snapper* is a clever little device that, on one side, plugs its input plug into an output socket, and, on the other side, exposes an output socket for plugging in a light or other device.

When a *Snapper* is in the ON state and is receiving power from its input plug, then the device connected to its output socket is receiving power as well. When you snap your fingers — making a clicking sound — any *Snapper* receiving power at the time of the snap toggles between the ON and OFF states.

In hopes of destroying the universe by means of a singularity, I have purchased n *Snapper* devices and chained them together by plugging the first one into a power socket, the second one into the first one, and so on. The light is plugged into the n-th *Snapper*.

Initially, all the *Snappers* are in the OFF state, so only the first one is receiving power from the socket, and the light is off. I snap my fingers once, which toggles the first *Snapper* into the ON state and gives power to the second one. I snap my fingers again, which toggles both *Snappers* and then promptly cuts power off from the second one, leaving it in the ON state, but with no power. I snap my fingers the third time, which toggles the first *Snapper* again and gives power to the second one. Now both *Snappers* are in the ON state, and if my light is plugged into the second *Snapper* it will be on.

I keep doing this for hours. Will the light be on or off after I have snapped my fingers k times? The light is on if and only if it's receiving power from the *Snapper* it's plugged into.

Input

The first line of the input gives a single integer t ($1 \leq t \leq 10000$) — the number of test cases.

Each of the following t lines contains two integers n and k ($1 \leq n \leq 30$, $0 \leq k \leq 10^{18}$).

Output

For each test case, output one line containing "`ON`" or "`OFF`", indicating the state of the light bulb.

Examples

standard input	standard output
4 1 0 1 1 4 0 4 47	OFF ON OFF ON

C. Colored Cubes

There are several colored cubes. All of them are of the same size but they may be colored differently. Each face of these cubes has a single color. Colors of distinct faces of a cube may or may not be the same.

Two cubes are said to be *identically colored* if some suitable rotations of one of the cubes give identical looks to both of the cubes. For example, two cubes shown in Figure 2 are identically colored. A set of cubes is said to be identically colored if every pair of them are identically colored.

A cube and its mirror image are not necessarily identically colored. For example, two cubes shown in Figure 3 are not identically colored.

You can make a given set of cubes identically colored by repainting some of the faces, whatever colors the faces may have. In Figure 4, repainting four faces makes the three cubes identically colored and repainting fewer faces will never do.

Your task is to write a program to calculate the minimum number of faces that needs to be repainted for a given set of cubes to become identically colored.

Input

The input is a sequence of datasets. A dataset consists of a header and a body appearing in this order. A header is a line containing one positive integer n and the body following it consists of n lines. You can assume that $1 \leq n \leq 4$. Each line in a body contains six color names separated by a space. A color name consists of a word or words connected with a hyphen (-). A word consists of one or more lowercase letters. You can assume that a color name is at most 24-characters long including hyphens.

A dataset corresponds to a set of colored cubes. The integer n corresponds to the number of cubes. Each line of the body corresponds to a cube and describes the colors of its faces. Color names in a line is ordered in accordance with the numbering of faces shown in Figure 5. A line

$$color_1 \ color_2 \ color_3 \ color_4 \ color_5 \ color_6$$

corresponds to a cube colored as shown in Figure 6.

The end of the input is indicated by a line containing a single zero. It is not a dataset nor a part of a dataset.

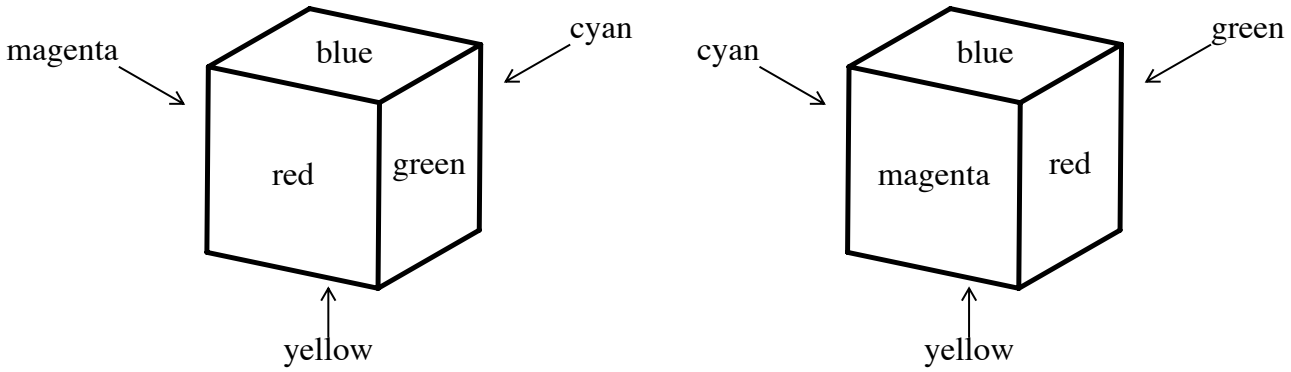

Figure 2: Identically colored cubes

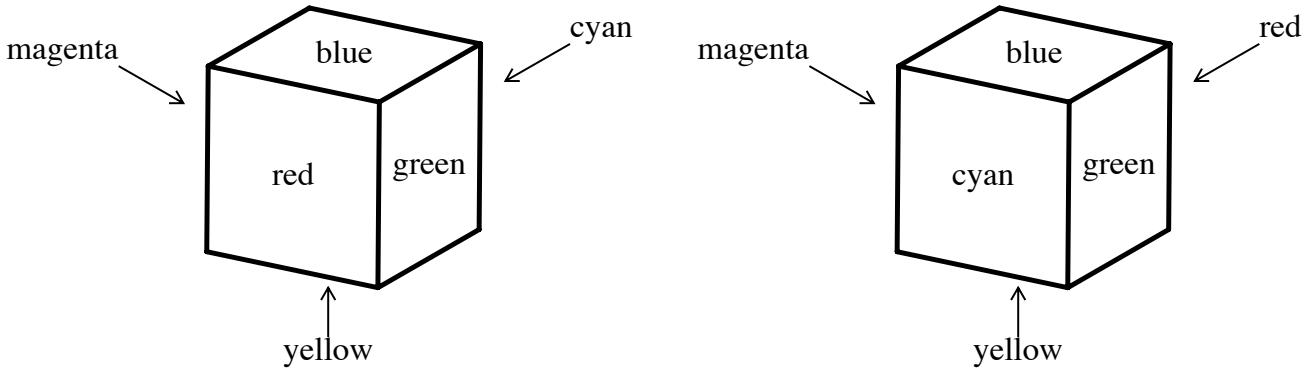

Figure 3: cubes that are not identically colored

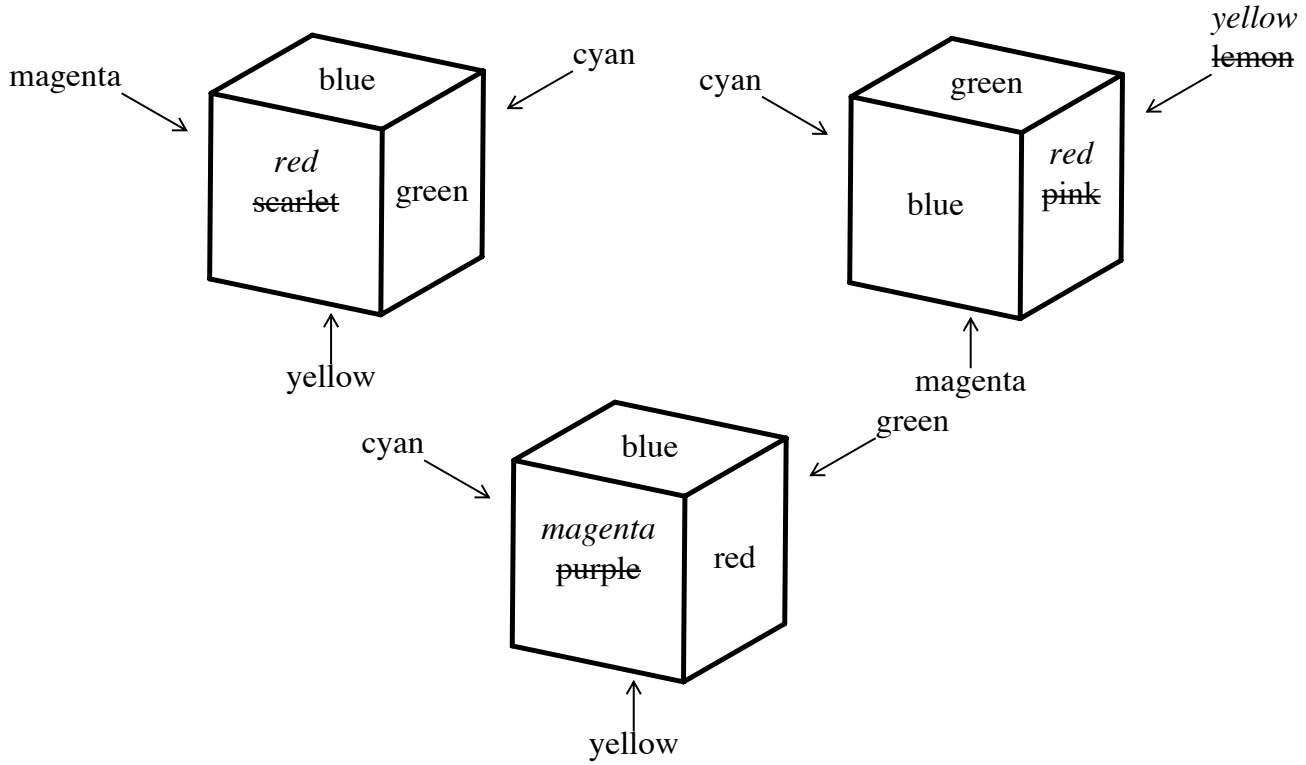

Figure 4: An example of recoloring

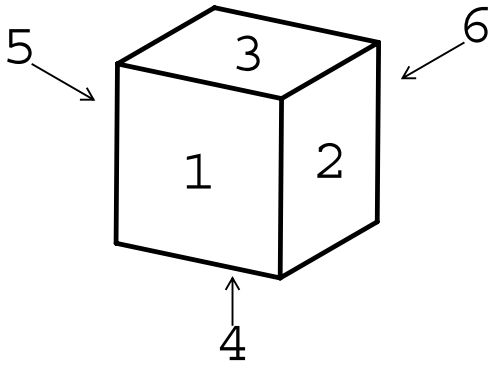

Figure 5: Numbering of faces

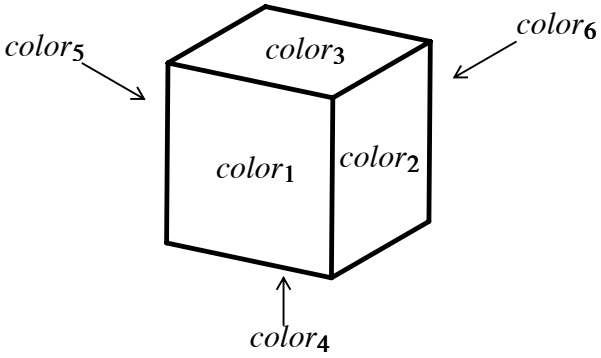

Figure 6: Coloring

Output

For each dataset, output a line containing the minimum number of faces that need to be repainted to make the set of cubes identically colored.

Sample Input

```
3
scarlet green blue yellow magenta cyan
blue pink green magenta cyan lemon
purple red blue yellow cyan green
2
red green blue yellow magenta cyan
cyan green blue yellow magenta red
2
red green gray gray magenta cyan
cyan green gray gray magenta red
2
red green blue yellow magenta cyan
magenta red blue yellow cyan green
3
red green blue yellow magenta cyan
cyan green blue yellow magenta red
magenta red blue yellow cyan green
3
blue green green green green blue
green blue blue green green green
green green green green green sea-green
3
red yellow red yellow red yellow
red red yellow yellow red yellow
red red red red red red
4
violet violet salmon salmon salmon salmon
```

```
violet salmon salmon salmon salmon violet
violet violet salmon salmon violet violet
violet violet violet violet salmon salmon
1
red green blue yellow magenta cyan
4
magenta pink red scarlet vermilion wine-red
aquamarine blue cyan indigo sky-blue turquoise-blue
blond cream chrome-yellow lemon olive yellow
chrome-green emerald-green green olive vilidian sky-blue
0
```

Output for the Sample Input

```
4
2
0
0
2
3
4
4
0
16
```

D. Strange Queries

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

You are given an array with n integers $a_1, a_2, ..., a_n$, and q queries to answer.

Each query consists of four integers l_1, r_1, l_2 and r_2. Your task is to count the number of pairs of indices (i, j) satisfying the following conditions:

1. $a_i = a_j$
2. $l_1 \le i \le r_1$
3. $l_2 \le j \le r_2$

Input

The first line of the input contains an integer n $(1 \le n \le 50\,000)$ — the size of the given array.

The second line contains n integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le n)$.

The third line contains an integer q $(1 \le q \le 50\,000)$ — the number of queries.

Each of the next q lines contains four integers l_1, r_1, l_2, r_2 $(1 \le l_1 \le r_1 \le n, 1 \le l_2 \le r_2 \le n)$, describing one query.

Output

For each query count the number of sought pairs of indices (i, j) and print it in a separate line.

Examples

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

output
1 2 5 6 6 2 2 0

E GukiZ Height

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

GukiZ loves hiking in the mountains. He starts his hike at the height 0 and he has some goal to reach, initially equal to h.

The mountains are described by a sequence of integers $A_0, A_1, ..., A_{n-1}$. In the first day GukiZ will change his height by A_0, in the second day by A_1, and so on. Mountains are a regular structure so the pattern repeats after the first n days. In general, in the i-th day GukiZ will change his height by $A_{(i-1)\%n}$.

Additionally, GukiZ will become more and more tired and in the i-th day his goal will decrease by i. For example, after first three days his goal will be equal to $h - 1 - 2 - 3 = h - 6$.

Note that A may contain negative elements, what represents going down from some hill. Moreover, GukiZ's height may become negative, and so does his goal!

You can assume that both GukiZ's height and goal change at the same moment (immediately and simultaneously) in the middle of a day.

Once GukiZ is at the height not less than his goal, he ends his hike. Could you calculate the number of days in his hike?

Input

The first line of the input contains two integers n and h ($1 \le n \le 10^5$, $1 \le h \le 10^9$) — the length of the array A and the initial goal, respectively.

The second line contains n integers $A_0, A_1, ..., A_{n-1}$ ($-10^9 \le A_i \le 10^9$).

Output

In a single line print the number of days in the GukiZ's hike.

It can be proved that for any valid input the answer exists.

Examples

input
3 45 7 -4 5

output
7

input
1 1 0

output
1

Note

GukiZ starts at height 0 with goal 45. We can describe his hike as follows:

1. After the first day he is at height 7 and his goal is 44.
2. Height 3, goal 42.
3. Height 8, goal 39.
4. Height 15, goal 35.
5. Height 11, goal 30.
6. Height 16, goal 24.
7. Height 23, goal 17.

After the 7-th day Gukiz's height is not less than his goal so he ends his hike.