This is the **problem set for the midterm contest**.

This problem set has **13 problems**. The items in the problem list below are links to each problem. You can click a problem name to jump to its page in the PDF file.

Problems are not in order of difficulty.

Verdict Information

Yes/Accepted: Your program passed all the judge's test cases.
Runtime Error: Your program produced an error while running.
Wrong Answer: Your program's output did not match the judge's output file.
Time-limit Exceeded: Your program was not able to process all input and terminate in a reasonable amount of time. You may have infinite loops.
Please Send to Correct Problem: Your program appears to be a solution for another problem that you intended to submit to.

Problem A: Range Sum Query

At the Association of Super Pretty Coders (ASPC), you've been tasked with manipulating data for who-knows-what purpose. You were asked by your superior to code a simple **range sum query**. Given an list of numbers, a range sum query provides the sum of all the numbers from one index A to another index B, where A is less than or equal to B.

For example, given the following list:

data[0]	data[1]	data[2]	data[3]	data[4]	data[5]	data[6]
3	4	5	2	3	4	1

examples of range sum queries are as follows:

data[0]	data[1]	data[2]	data[3]	data[4]	data[5]	data[6]
3	4	5	2	3	4	1

The result of a range sum query from 1 to 3 is 11, since $4 + 5 + 2 = 11$.

data[0]	data[1]	data[2]	data[3]	data[4]	data[5]	data[6]
3	4	5	2	3	4	1

The result of a range sum query from 4 to 5 is 7, since $3 + 4 = 7$.

Input

The first line of input contains a single integer N $(1 \leq N \leq 100)$, the number of elements in the list.

The second line of input contains N integers separated by spaces: the list. All integers in the list range from 1 to 100.

The third line of input contains two integers A and B $(0 \leq A \leq B < N)$.

This represents a single range sum query from index A to index B. Indices start at zero.

Output

Your program should output

```
RSQ(A,B)=X
```

where A and B are the indices of the range sum query, and X is the value of the range sum query.

Sample Input #1

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

Sample Output #1

```
RSQ(1,3)=11
```

Sample Input #2

```
3
56 23 12
1 2
```

Sample Output #2

```
RSQ(1,2)=35
```

Problem B: Countdown

The Tetris game Aldrich is working on is still missing a few minor features!

Aldrich has been too busy to create a proper countdown system.
He just needs the server to continuously broadcast lines of text to the clients, so that each line counts down from one number to another.

Can you help him (again)?

Input
Input consists of two integers A and B $(0 \leq A \leq B \leq 100)$.

Output
Output all the integers from A to B, starting from B going down to A.
Each number should be on its own line.

Sample Input #1
```
1 5
```

Sample Output #1
```
5
4
3
2
1
```

Sample Input #2
```
3 8
```

Sample Output #2
```
8
7
6
5
4
3
```

Problem C: Cancellable Quiz

Your friend is currently in a Computer Science class. His teacher gave N quizzes, each worth 10 points. The class is really hard, so his teacher decided to *cancel* everyone's lowest quiz.

Example

If the teacher gave 3 quizzes, and your friend's scores were $^{10}/_{10}$, $^{5}/_{10}$, $^{9}/_{10}$ then normally his total quiz score would be $^{24}/_{30}$.

If you were to cancel your friend's lowest quiz, we would ignore the $^{5}/_{10}$.

After cancelling the lowest quiz, your friend's total score would be $^{19}/_{20}$.

Note: If there are 2 or more lowest quizzes, only one of them will be cancelled.

Input

The first line of input contains a single integer $(2 \leq N \leq 20)$, the number of quizzes.

The second line of input contains N integers, $q_1, q_2, \cdots, q_N$, which are your friend's scores in the quizzes. (q_1 is your friend's score for quiz 1, q_2 is your friend's score for quiz 2, etc.) The quiz scores will range from 0 to 10.

Output

Output two lines.

```
Raw Score=X/A
Final Score=Y/B
```

where:

- X is your friend's total score before his lowest quiz was cancelled.
- A is the highest possible score before cancelling the lowest quiz.
- Y is your friend's total score after cancelling his lowest quiz.
- B is the highest possible score after cancelling one quiz.

Sample Input #1

```
5
9 10 8 5 10
```

Sample Output #1

```
Raw Score=42/50
Final Score=37/40
```

Sample Input #2

```
3
10 9 9
```

Sample Output #2

```
Raw Score=28/30
Final Score=19/20
```

Explanation for sample input #1.

All of the quizzes are worth 10 points, so your friend's total score is

$$\frac{(9 + 10 + 8 + 5 + 10)}{50} = \mathbf{42/50}$$

Note that we do not need to simplify the fraction. Your friend's score after cancelling his lowest quiz $\left(^{5}/_{10}\right)$ would be:

$$\frac{(9 + 10 + 8 + 10)}{40} = \mathbf{37/40}$$

Problem D: Printing Diamonds

Nicko loves diamonds. However, diamonds tend to be really expensive, so he settles with just looking at diamonds in his computer screen. One day, he opens his internet browser only to find out that his internet connection is not working, and thus he cannot search for pictures of his beloved diamonds using Moogle.

In desperation, Nicko decides to seek for your help to print diamonds in his computer screen using the asterisk (*) character. By printing the * character in multiple lines, with each line having different number of * characters, it is possible to print a diamond onto the screen. However, he is very particular about the size of the diamonds that he wants to see. Thus, he gives you a number N, where N is the row number with the most number of asterisks.
As an example, refer to the figures below:

```
1    *
```

```
1     *
2    ***
3     *
```

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

For the first diamond, N is 1, since it is a diamond with only one row. For the second diamond, N is 2, since the 2nd row is the row where the diamond has the most number of asterisks. For the 3rd diamond, N is 3. Of course, Nicko wants to see a lot of diamonds, so he gives you multiple values of this number N.

Input
The first line of input contains an integer T $(1 \leq T \leq 10)$, the number of diamonds you should print.
This is followed by T lines with each line containing the number N $(1 \leq N \leq 30)$, the size of a diamond you should print.

Output
Print the diamonds for each of the values of N in the input.
The diamonds should be printed consecutively, with no extra lines or spaces.

Do not print any trailing spaces (i.e. extra spaces at the end of each line).
There are no spaces in between the asterisks.
If you have extra spaces in your output, your program will be marked wrong.

Sample Input

```
3
1
2
3
```

Sample Output

```
*
 *
***
 *
  *
 ***
*****
 ***
  *
```

Sample Output (with all spaces represented by a zero for clarity)
Do NOT print zeroes in the output of your program.

```
*
0*
***
0*
00*
0***
*****
0***
00*
```

Problem E: User Accounts

Aldrich needs help in setting up the user accounts for the midterm contest. He wants to generate accounts with the following username format:

`teamX`

where X is the team number.

He only has to make at most 99 accounts, and he wants to make all usernames the same length. Many account names are 6 characters long:

`team23, team41, team10, etc.`

But when the team number is a single digit number, we have account names which are 5 characters long!

`team1, team5, team7, etc.`

One way to fix this is to add **leading zeros** to the usernames of teams with a username which is too short:

`team1` → `team01`
`team5` → `team05`
`team7` → `team07`

Can you implement a program which ensures that all team names are 6 characters long?

Input

Input consists of two integers A and B $(1 \leq A \leq B \leq 99)$.

Output

Output all the team usernames from team A to team B, in order.

Sample Input #1
```
1 10
```
Sample Output #1
```
team01
team02
team03
team04
team05
team06
team07
team08
team09
team10
```

Sample Input #2
```
9 13
```
Sample Output #2
```
team09
team10
team11
team12
team13
```

Sample Input #3
```
51 53
```
Sample Output #3
```
team51
team52
team53
```

Problem F: Random Generator

One simple method to generate random numbers is using a **linear congruential generator (LCG).** The sequence of random numbers generated by an LCG is determined by the following equation.

$$X_{n+1} = (aX_n + c) \mod m$$

Note: $a \mod b$ is the remainder when you divide a by b.

Usually, one picks values for a, c and m, and an initial value X_0.
Then X_1 is determined by plugging in the values a, c, m and X_0 into the equation.
Then, one gets another number, X_2 by plugging in a, c, m and X_1.
This can continue infinitely: $X_0, X_1, X_2, X_3, \cdots$ forms a sequence of numbers which look random.

Example
If $a = 2017, c = 5, m = 1007$ and $X_0 = 1$, then we can get X_1 like this:

$$X_1 = (aX_0 + c) \mod m$$
$$X_1 = (2017 \times (1) + 5) \mod 1007$$
$$X_1 = (2022) \mod 1007$$
$$X_1 = 8$$

In this problem, **you only need to get one number** using an LCG.

Input
The input consists of four integers, a, c, m and X_0, in that order.
All numbers will be positive integers from 1 to 10000.

Output
Output X_1.

Sample Input #1
```
2017 5 1007 1
```

Sample Output #1
```
8
```

Sample Input #2
```
543 12 3127 3
```

Sample Output #2
```
1641
```

Sample Input #3
```
2 3 5 7
```

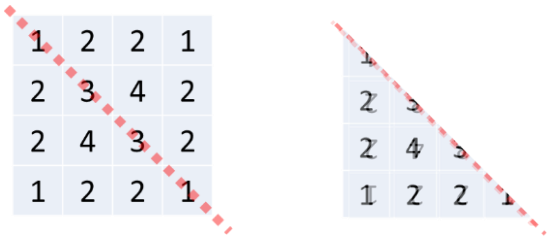
Sample Output #3
```
2
```

Problem G: Paper Folding

You've been asked to process an $N \times N$ grid of numbers, and you've been asked to check if the grid is **symmetric**.
For example, here is a 4×4 grid of numbers.

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

To check if a grid is symmetric, **you draw a line from the upper-left corner to the lower-right corner**.



This divides the grid into two triangles. If the two triangles are **mirror images** of each other, then the grid is symmetric. In other words, if you **fold the grid along the line**, then the numbers in each triangle will match up.

Input
The first line of input contains an integer N $(1 \leq N \leq 10)$.
N lines follow, each line containing N integers separated by spaces.

Output
Output `Yes` if the grid is symmetric, and `No` if it is not symmetric.

Sample Input #1
```
4
1 2 2 1
2 4 3 2
2 3 4 2
1 2 2 1
```

Sample Output #1
```
Yes
```

Sample Input #2
```
3
1 2 3
4 5 6
7 8 9
```

Sample Output #2
```
No
```

Sample Input #3

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

Sample Output #3

```
No
```

Sample Input #4

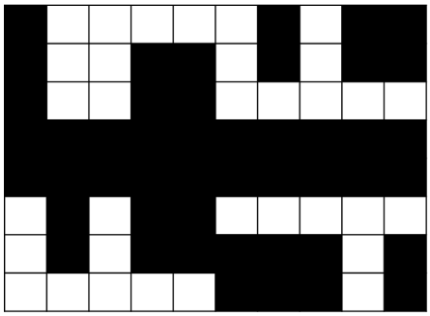
```
3
1 1 1
1 1 1
1 1 1
```

Sample Output #4

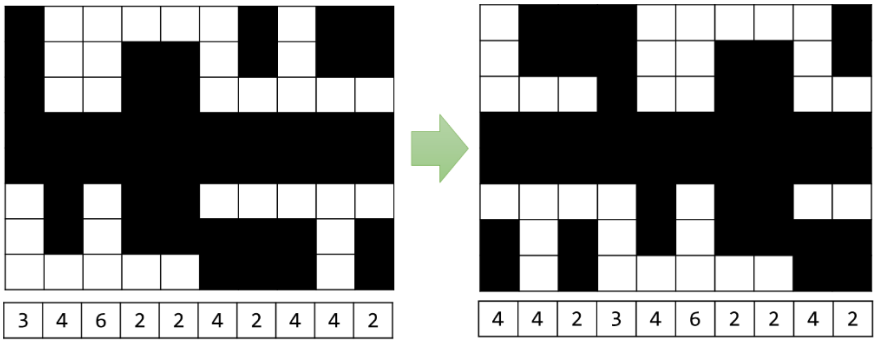
```
Yes
```

Problem H: Screen Scrolling

Your friend Nina is making a game where you have to move a black-and-white grid. She'd like to keep track of how many cells are white when the screen is scrolled. To make things easier to count, she divides her screen into C columns, and counts the number of pixels which are lit up in each column.
Below is a case where $C = 10$.



She wants to scroll the screen k units to the right. In her game, the screen wraps around horizontally. This means that if a part of the screen goes past the right side, it will appear back on the left side. Below is a case where $C = 10$ and $k = 3$. This means there are 10 columns and the screen is moved 3 columns to the right.



Input

The first line of input contains two integers C and k $(1 \leq k \leq C \leq 100)$
The second line of input contains C integers $p_1, p_2, p_3, \cdots p_N$, where p_i denotes the number of pixels in column i which are lit up before the screen is scrolled.

Output

Output C integers, where the ith integer is the number of pixels in column i which are lit up **after** the screen is scrolled. Output each integer on a single line.

Sample Input #1

```
10 3
3 4 6 2 2 4 2 4 4 2
```

Sample Output #1

```
4
4
2
3
4
6
2
2
4
2
```

Sample Input #2

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

Sample Output #2

```
2
3
4
5
1
```

Sample Input #3

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

Sample Output #3

```
1
2
3
4
5
```

Problem I: Is it Sorted Again?

Your professor returned your second draft of your research paper. It seems that it would be better if you used more than just 3 data points. You have all your results in an list of integers. Now you just have to analyze them.

Remember:
A sequence is **increasing** if every number in the sequence is greater than or equal to the number before it.
A sequence is **decreasing** if every number in the sequence is less than or equal to the number before it.

Input
The first line of input contains a single integer N $(1 \leq N \leq 100)$, the number of elements in the list.
The second line of input contains N integers separated by spaces: the list of integers. All integers in the integer array range from -100 to 100.

Output
Output a single line:

- `equal` if all the elements in the sequence are the same
- `increasing` if the sequence is increasing, and the elements in the sequence are not all equal
- `decreasing` if the sequence is decreasing, and the elements in the sequence are not all equal
- `not sorted` if none of the above conditions apply

Sample Input #1
```
5
1 3 5 7 9
```

Sample Output #1
```
increasing
```

Sample Input #2
```
6
4 4 3 3 -1 -1
```

Sample Output #2
```
decreasing
```

Sample Input #3
```
3
2 2 2
```

Sample Output #3
```
equal
```

Sample Input #4
```
8
1 2 3 1 2 3 1 4
```

Sample Output #4
```
not sorted
```

Problem J: Convert

Given a temperature in Celsius, convert it to Fahrenheit, where

$$T_F = \left(T_C \times \frac{9}{5}\right) + 32$$

where T_F denotes the temperature in Fahrenheit and T_C denotes the temperature in Celsius.

Input
Input consists of only one line. The line contains the number T_C $(-100 \leq T_C \leq 100)$, a temperature in Celsius.

Output
Print T_F, the temperature in Fahrenheit.
Your program will be judged as correct as long as it is correct to up to 3 decimal places.

Sample Input #1	Sample Output #1
`-10`	`14.0`
Sample Input #2	**Sample Output #2**
`1.5`	`34.7`
Sample Input #3	**Sample Output #3**
`50.3`	`122.53999999999999`

Problem K: Information Tracker

Aldrich plans to eat out for lunch today. He wants to eat out with friends. He has N friends, and he told his first friend that he plans on eating out. To make this problem easier, we number Aldrich's friends with integers from 1 to N.

But today is a very busy day! Not all of Aldrich's friends will get to see each other. It turns out that today, M pairs of people will meet. When two of Aldrich's friends meet, if one of them knows Aldrich's plans, then he'll tell the other friend, so that they will both know that Aldrich plans on eating out.

Input
The first line of input contains two numbers N and M $(1 \leq N, M \leq 500)$. N is the number of friends Aldrich has, and M is the number of meetings that take place.

M lines follow, each containing two integers a and b $(1 \leq a, b \leq N; a \neq b)$, the two people who meet.

The meetings all happen in order (i.e. the first meeting in the list is the first meeting that happens, the next meeting in the list happens next, and so on.) Remember that before all the meetings take place, friend 1 is the only friend who knows Aldrich's plans.

Output
Output the number of friends who know Aldrich's plans after the M meetings.

Sample Input #1
```
5 3
1 2
3 2
4 5
```

Sample Output #1
```
3
```

Sample Input #2
```
4 5
2 4
1 2
2 4
4 3
3 4
```

Sample Output #2
```
4
```

Explanations for the sample inputs are on the next page.

Explanation for sample input 1:

In the first sample input, Aldrich has 5 friends, and there are 3 meetings. At the start, only friend 1 knows about Aldrich's plans.

At the first meeting, friends 1 and 2 meet. Friend 1 tells friend 2 about Aldrich's plans, so the set of people who know Aldrich's plans is now $\{1,2\}$.

At the second meeting, friends 3 and 2 meet. Friend 2 tells friend 3 about Aldrich's plans, so the set of people who know Aldrich's plans is now $\{1,2,3\}$.

At the third meeting, friends 4 and 5 meet. None of them know about Aldrich's plans, so the set of people who know Aldrich's plans is still $\{1,2,3\}$.

After all the meetings take place, 3 of Aldrich's friends know about his plans: friends 1, 2, and 3.

Explanation for sample input 2:

In the second sample input, Aldrich has 4 friends, and there are 5 meetings. At the start, only friend 1 knows about Aldrich's plans.

At the first meeting, friends 2 and 4 meet. None of them know about Aldrich's plans, so the set of people who know Aldrich's plans is still $\{1\}$.

At the second meeting, friends 1 and 2 meet. Friend 1 tells friend 2 about Aldrich's plans, so the set of people who know Aldrich's plans is now $\{1,2\}$.

At the third meeting, friends 2 and 4 meet again. Friend 2 tells friend 4 about Aldrich's plans, so the set of people who know Aldrich's plans is now $\{1,2,4\}$.

At the fourth meeting, friends 4 and 3 meet. Friend 4 tells friend 3 about Aldrich's plans, so the set of people who know Aldrich's plans is now $\{1,2,3,4\}$.

At the fifth meeting, friends 3 and 4 meet again, so nothing happens.

After all the meetings take place, 4 of Aldrich's friends know about his plans: friends 1, 2,3, and 4.

Problem L: Lower or Higher?

Well, we thought of giving you this problem:

- Input consists of two numbers A and B. The two numbers are not going to be equal.
- If $A < B$ output `"A is lower than B"`.
- Otherwise, if $A > B$, output `"A is higher than B"`.

But don't you think that problem is too easy?

We tried giving that problem in another contest, and we have a lot of submissions.

Can you check the submissions for us?

Input
The first line of input contains the number of test cases T $(1 \leq T \leq 200)$.

T lines follow, each describing a single test case. Each test case is described by a line containing one of the following

`"A is lower than B"`

`"A is higher than B"`

where A and B are positive integers $(1 \leq A, B \leq 500000)$.

Output
For each test case, output `"CORRECT"` if the input is correct (it's correct that A is higher/lower than B), and `"WRONG"` if it is not.

Sample Input #1
```
3
3 is lower than 5
4 is higher than 7
7 is higher than 9
```

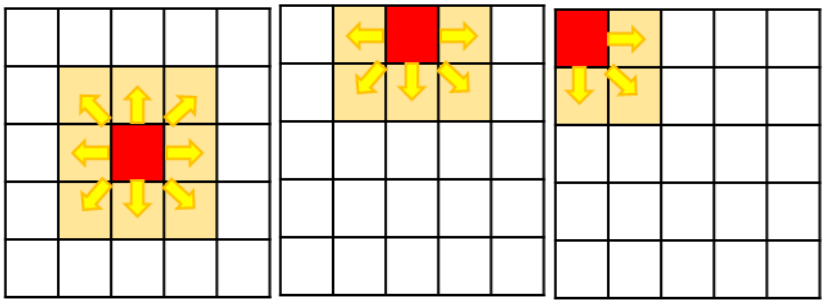
Sample Output #1
```
CORRECT
WRONG
WRONG
```

Sample Input #2
```
5
120 is lower than 420
90 is lower than 99
50 is higher than 50
1 is lower than 1
40 is higher than 20
```

Sample Output #2
```
CORRECT
CORRECT
WRONG
WRONG
CORRECT
```

Problem M: Conway's Game of Life

Conway's Game of Life is a simulation run on a 2D grid.
Each cell in the grid is either **alive** or **dead**.



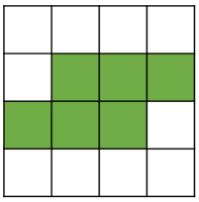
The neighbors of a cell are the cells surrounding it in all eight directions. In the drawing to the left, the yellow cells indicate the neighbors of the red cell.

Cells in the corners have 3 neighbors, and cells on the sides have 5 neighbors.
After one unit of time, the cells in the grid change according to the following rules:

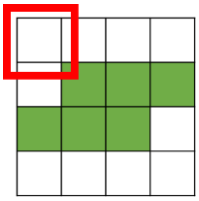
- Any alive cell with fewer than two alive neighbors dies.
- Any alive cell with two or three alive neighbors remains alive.
- Any alive cell with more than three alive neighbors dies.
- Any dead cell with exactly three alive neighbors becomes alive; if a dead cell does not have three neighbors, it remains dead.

Example

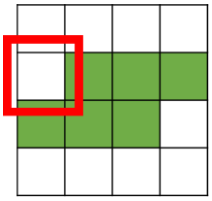
Consider the following grid. *Green cells indicate alive cells, and white cells indicate dead cells.*



We can look at what happens to each cell one by one.
For example, look at the cell highlighted by the red box:

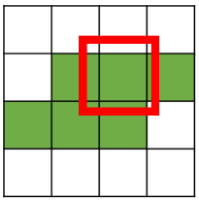


This dead cell has 3 neighbors, but only 1 alive neighbor. This means that this cell will remain **dead**.
On the other hand, look at this cell:



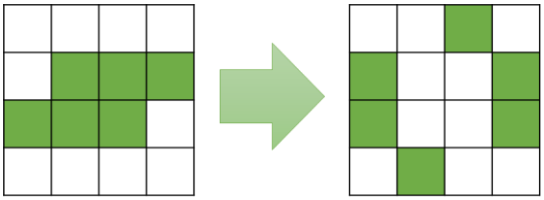
This dead cell has 5 neighbors, but exactly 3 alive neighbors. This means that this cell will become **alive**.

Finally, look at this cell:



This alive cell has 8 neighbors, and has more than 3 alive neighbors. This means that this cell will become **dead**.

By checking all the cells in the grid, we can see that the grid will turn into the following:



Write a program which determines how many cells are live after one time unit.

Input

The first line of input contains two positive integers R ($1 \leq \mathrm{R} \leq 500$) and C ($1 \leq \mathrm{C} \leq 500$).
R lines follow, each line containing C integers. This represents the initial state of the grid.
`1` denotes a live cell, and `0` denotes a dead cell.

Output

Print out a single line "`Alive cells: x`" (without the quotes) where x is an integer denoting the number of live cells after one time unit.

Sample Input #1

```
5 6
0 0 0 0 0 0
0 1 0 0 0 0
0 0 1 1 0 0
0 1 1 0 0 0
0 0 0 0 0 0
```

Sample Output #1

```
Alive cells: 5
```

Sample Input #2

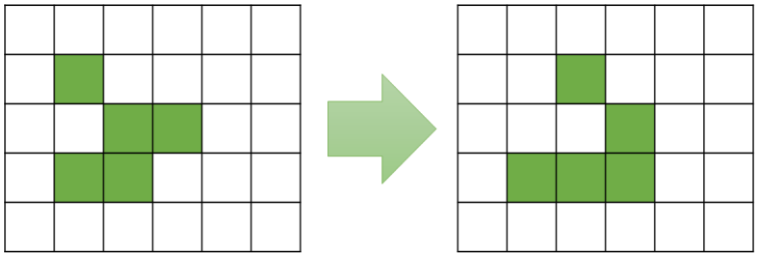
```
4 4
1 1 0 0
1 1 0 0
0 0 1 1
0 0 1 1
```

Sample Output #2

```
Alive cells: 6
```

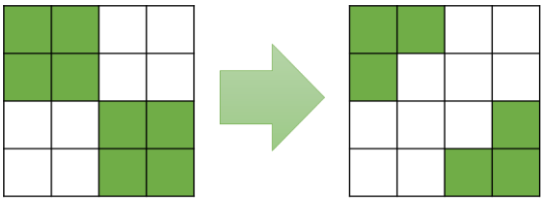
Explanations for the sample inputs are on the next page.

Explanation for the first sample case:



There are 5 alive cells after one unit of time.

Explanation for the second sample case:



There are 6 alive cells after one unit of time.