A. Gregor and Cryptography

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

Gregor is learning about RSA cryptography, and although he doesn't understand how RSA works, he is now fascinated with prime numbers and factoring them.

Gregor's favorite **prime** number is P. Gregor wants to find two *bases* of P. Formally, Gregor is looking for two integers a and b which satisfy both of the following properties.

- $P \bmod a = P \bmod b$, where $x \bmod y$ denotes the remainder when x is divided by y, and
- $2 \le a < b \le P$.

Help Gregor find two bases of his favorite prime number!

Input

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

Each subsequent line contains the integer P ($5 \le P \le 10^9$), with P guaranteed to be prime.

Output

Your output should consist of t lines. Each line should consist of two integers a and b ($2 \le a < b \le P$). If there are multiple possible solutions, print any.

Example

input

```
2
17
5
```

output

```
3 5
2 4
```

Note

The first query is $P = 17$. $a = 3$ and $b = 5$ are valid *bases* in this case, because $17 \bmod 3 = 17 \bmod 5 = 2$. There are other pairs which work as well.

In the second query, with $P = 5$, the only solution is $a = 2$ and $b = 4$.

B. Gregor and the Pawn Game

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

There is a chessboard of size n by n. The square in the i-th row from top and j-th column from the left is labelled (i, j).

Currently, Gregor has some pawns in the n-th row. There are also enemy pawns in the 1-st row. On one turn, Gregor moves one of **his** pawns. A pawn can move one square up (from (i, j) to $(i-1, j)$) if there is no pawn in the destination square. Additionally, a pawn can move one square diagonally up (from (i, j) to either $(i-1, j-1)$ or $(i-1, j+1)$) if and only if there is an enemy pawn in that square. The enemy pawn is also removed.

Gregor wants to know what is the maximum number of his pawns that can reach row 1?

Note that only Gregor takes turns in this game, and **the enemy pawns never move**. Also, when Gregor's pawn reaches row 1, it is stuck and cannot make any further moves.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

Each test case consists of three lines. The first line contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the size of the chessboard.

The second line consists of a string of binary digits of length n, where a 1 in the i-th position corresponds to an enemy pawn in the i-th cell from the left, and 0 corresponds to an empty cell.

The third line consists of a string of binary digits of length n, where a 1 in the i-th position corresponds to a Gregor's pawn in the i-th cell from the left, and 0 corresponds to an empty cell.

It is guaranteed that the sum of n across all test cases is less than $2 \cdot 10^5$.

Output

For each test case, print one integer: the **maximum** number of Gregor's pawns which can reach the 1-st row.

Example

input

```
4
3
000
111
4
1111
1111
3
010
010
5
11001
00000
```

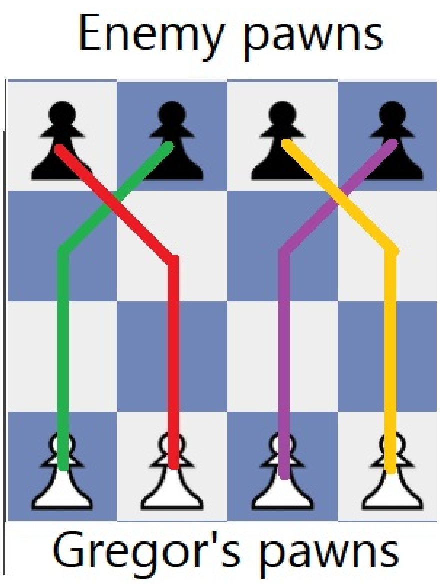
output

```
3
4
0
0
```

Note

In the first example, Gregor can simply advance all 3 of his pawns forward. Thus, the answer is 3.

In the second example, Gregor can guarantee that all 4 of his pawns reach the enemy row, by following the colored paths as demonstrated in the diagram below. Remember, only Gregor takes turns in this "game"!



In the third example, Gregor's only pawn is stuck behind the enemy pawn, and cannot reach the end.

In the fourth example, Gregor has no pawns, so the answer is clearly 0.

C. Web of Lies

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

When you play the game of thrones, you win, or you die. There is no middle ground.

Cersei Lannister, *A Game of Thrones* by George R. R. Martin

There are n nobles, numbered from 1 to n. Noble i has a power of i. There are also m "friendships". A friendship between nobles a and b is always mutual.

A noble is defined to be *vulnerable* if both of the following conditions are satisfied:

- the noble has at least one friend, and
- **all** of that noble's friends have a higher power.

You will have to process the following three types of queries.

1. Add a friendship between nobles u and v.
2. Remove a friendship between nobles u and v.
3. Calculate the answer to the following process.

The process: all vulnerable nobles are simultaneously killed, and all their friendships end. Then, it is possible that new nobles become vulnerable. The process repeats itself until no nobles are vulnerable. It can be proven that the process will end in finite time.

After the process is complete, you need to calculate the number of remaining nobles.

Note that the results of the process are **not** carried over between queries, that is, every process starts with all nobles being alive!

Input

The first line contains the integers n and m ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$) — the number of nobles and number of original friendships respectively.

The next m lines each contain the integers u and v ($1 \le u, v \le n$, $u \ne v$), describing a friendship. No friendship is listed twice.

The next line contains the integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

The next q lines contain the queries themselves, each query has one of the following three formats.

- $1\ u\ v$ ($1 \le u, v \le n$, $u \ne v$) — add a friendship between u and v. It is guaranteed that u and v are not friends at this moment.
- $2\ u\ v$ ($1 \le u, v \le n$, $u \ne v$) — remove a friendship between u and v. It is guaranteed that u and v are friends at this moment.
- 3 — print the answer to the process described in the statement.

Output

For each type 3 query print one integer to a new line. It is guaranteed that there will be at least one type 3 query.

Examples

input

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

output

```
2
1
```

input

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

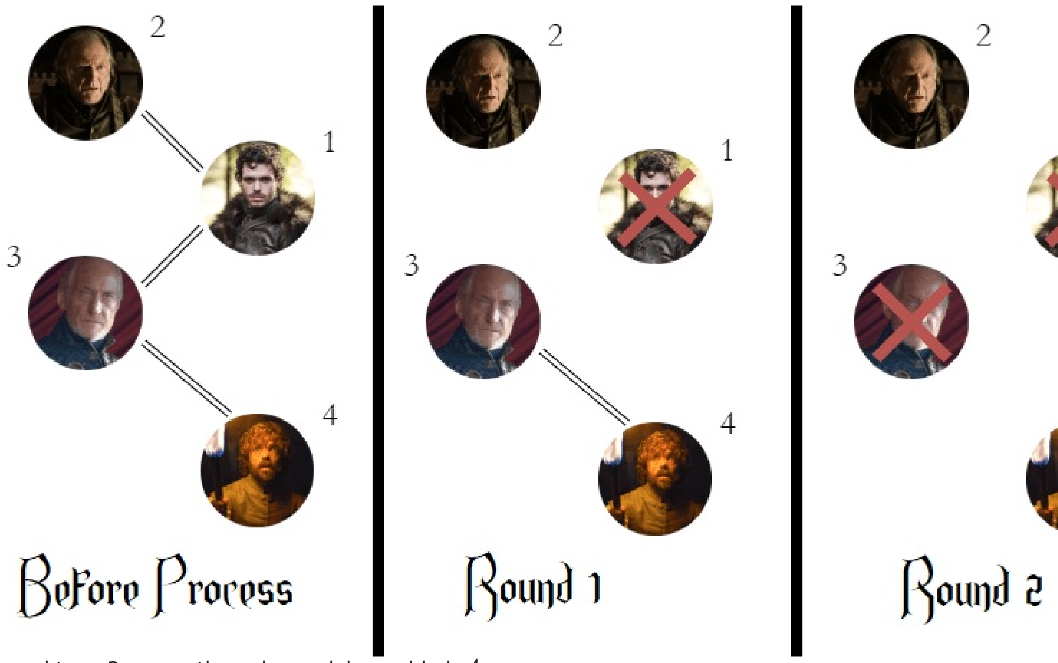
output

```
1
```

Note

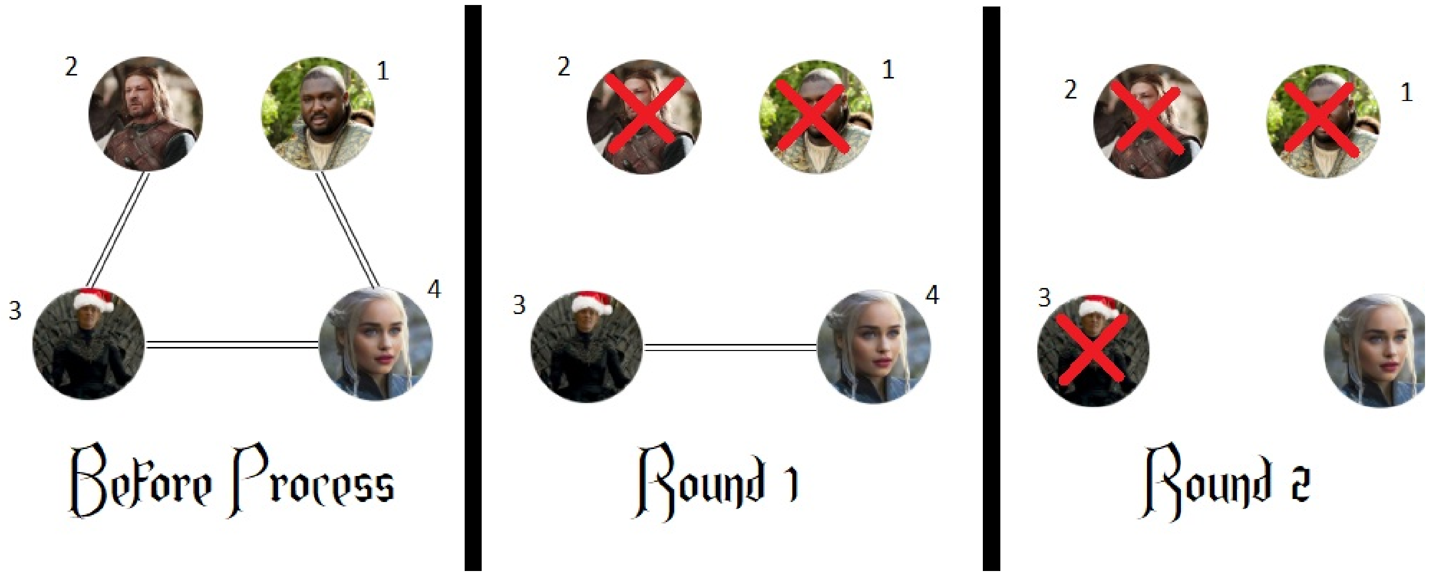
Consider the first example. In the first type 3 query, we have the diagram below.

In the first round of the process, noble 1 is weaker than all of his friends (2 and 3), and is thus killed. No other noble is vulnerable in round 1. In round 2, noble 3 is weaker than his only friend, noble 4, and is therefore killed. At this point, the process ends, and the answer is 2.



In the second type 3 query, the only surviving noble is 4.

The second example consists of only one type 3 query. In the first round, two nobles are killed, and in the second round, one noble is killed. The final answer is 1, since only one noble survives.



D. Integers Have Friends

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

British mathematician John Littlewood once said about Indian mathematician Srinivasa Ramanujan that "every positive integer was one of his personal friends."

It turns out that positive integers can also be friends with each other! You are given an array a of distinct positive integers.

Define a **subarray** $a_i, a_{i+1}, \ldots, a_j$ to be a *friend group* if and only if there exists an integer $m \geq 2$ such that $a_i \bmod m = a_{i+1} \bmod m = \ldots = a_j \bmod m$, where $x \bmod y$ denotes the remainder when x is divided by y.

Your friend Gregor wants to know the size of the largest friend group in a.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 2 \cdot 10^4$).

Each test case begins with a line containing the integer n ($1 \leq n \leq 2 \cdot 10^5$), the size of the array a.

The next line contains n positive integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^{18}$), representing the contents of the array a. It is guaranteed that all the numbers in a are **distinct**.

It is guaranteed that the sum of n over all test cases is less than $2 \cdot 10^5$.

Output

Your output should consist of t lines. Each line should consist of a single integer, the size of the largest friend group in a.

Example

input

```
4
5
1 5 2 4 6
4
8 2 5 10
2
1000 2000
8
465 55 3 54 234 12 45 78
```

output

```
3
3
2
6
```

Note

In the first test case, the array is $[1, 5, 2, 4, 6]$. The largest friend group is $[2, 4, 6]$, since all those numbers are congruent to 0 modulo 2, so $m = 2$.

In the second test case, the array is $[8, 2, 5, 10]$. The largest friend group is $[8, 2, 5]$, since all those numbers are congruent to 2 modulo 3, so $m = 3$.

In the third case, the largest friend group is $[1000, 2000]$. There are clearly many possible values of m that work.

E. The Three Little Pigs

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

Three little pigs from all over the world are meeting for a convention! Every minute, a triple of 3 new pigs arrives on the convention floor. After the n-th minute, the convention ends.

The big bad wolf has learned about this convention, and he has an attack plan. At some minute in the convention, he will arrive and eat exactly x pigs. Then he will get away.

The wolf wants Gregor to help him figure out the number of possible attack plans that involve eating exactly x pigs for various values of x ($1 \le x \le 3n$). Two attack plans are considered different, if they occur at different times or if the sets of little pigs to eat are different.

Note that all queries are independent, that is, the wolf does not eat the little pigs, he only makes plans!

Input

The first line of input contains two integers n and q ($1 \le n \le 10^6$, $1 \le q \le 2 \cdot 10^5$), the number of minutes the convention lasts and the number of queries the wolf asks.

Each of the next q lines contains a single integer x_i ($1 \le x_i \le 3n$), the number of pigs the wolf will eat in the i-th query.

Output

You should print q lines, with line i representing the number of attack plans if the wolf wants to eat x_i pigs. Since each query answer can be large, output each answer modulo $10^9 + 7$.

Examples

input

```
2 3
1
5
6
```

output

```
9
6
1
```

input

```
5 4
2
4
6
8
```

output

```
225
2001
6014
6939
```

Note

In the example test, $n = 2$. Thus, there are 3 pigs at minute 1, and 6 pigs at minute 2. There are three queries: $x = 1$, $x = 5$, and $x = 6$.

If the wolf wants to eat 1 pig, he can do so in $3 + 6 = 9$ possible attack plans, depending on whether he arrives at minute 1 or 2.

If the wolf wants to eat 5 pigs, the wolf cannot arrive at minute 1, since there aren't enough pigs at that time. Therefore, the wolf has to arrive at minute 2, and there are 6 possible attack plans.

If the wolf wants to eat 6 pigs, his only plan is to arrive at the end of the convention and devour everybody.

Remember to output your answers modulo $10^9 + 7$!

F1. Gregor and the Odd Cows (Easy)

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

*This is the easy version of the problem. The only difference from the hard version is that in this version all coordinates are **even**.*

There are n fence-posts at distinct coordinates on a plane. It is guaranteed that no three fence posts lie on the same line.

There are an infinite number of cows on the plane, one at every point with integer coordinates.

Gregor is a member of the Illuminati, and wants to build a triangular fence, connecting 3 distinct existing fence posts. A cow **strictly** inside the fence is said to be *enclosed*. If there are an **odd** number of enclosed cows and the area of the fence is an **integer**, the fence is said to be *interesting*.

Find the number of interesting fences.

Input

The first line contains the integer n ($3 \le n \le 6000$), the number of fence posts which Gregor can choose to form the vertices of a fence.

Each of the next n line contains two integers x and y ($0 \le x, y \le 10^7$, x and y are **even**), where (x, y) is the coordinate of a fence post. All fence posts lie at distinct coordinates. No three fence posts are on the same line.

Output

Print a single integer, the number of interesting fences. Two fences are considered different if they are constructed with a different set of three fence posts.

Examples

input

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

output

```
1
```

input

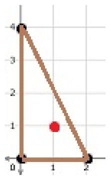
```
5
0 0
2 16
30 14
4 6
2 10
```

output

```
3
```

Note

In the first example, there is only 1 fence. That fence is interesting since its area is 4 and there is 1 enclosed cow, marked in red.



In the second example, there are 3 interesting fences.

- $(0, 0)$ — $(30, 14)$ — $(2, 10)$
- $(2, 16)$ — $(30, 14)$ — $(2, 10)$
- $(30, 14)$ — $(4, 6)$ — $(2, 10)$

F2. Gregor and the Odd Cows (Hard)

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is the hard version of the problem. The only difference from the easy version is that in this version the coordinates can be ***both*** *odd and even.*

There are n fence-posts at distinct coordinates on a plane. It is guaranteed that no three fence posts lie on the same line.

There are an infinite number of cows on the plane, one at every point with integer coordinates.

Gregor is a member of the Illuminati, and wants to build a triangular fence, connecting 3 distinct existing fence posts. A cow **strictly** inside the fence is said to be *enclosed*. If there are an **odd** number of enclosed cows and the area of the fence is an **integer**, the fence is said to be *interesting*.

Find the number of interesting fences.

Input

The first line contains the integer n ($3 \le n \le 6000$), the number of fence posts which Gregor can choose to form the vertices of a fence.

Each of the next n line contains two integers x and y ($0 \le x, y \le 10^7$, where (x, y) is the coordinate of a fence post. All fence posts lie at distinct coordinates. No three fence posts are on the same line.

Output

Print a single integer, the number of interesting fences. Two fences are considered different if they are constructed with a different set of three fence posts.

Examples

input

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

output

```
1
```

input

```
4
1 8
0 6
5 2
5 6
```

output

```
1
```

input

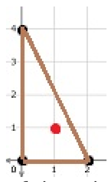
```
10
170 59
129 54
5 98
129 37
58 193
154 58
24 3
13 138
136 144
174 150
```

output

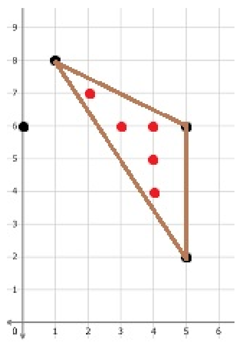
```
29
```

Note

In the first example, there is only 1 fence. That fence is interesting since its area is 4 and there is 1 enclosed cow, marked in red.



In the second example, there are 4 possible fences. Only one of them is interesting however. That fence has an area of 8 and 5 enclosed cows.



9
8
7
6
5
4
3
2
1
0
1
2
3
4
5
6