Problem A

The One With Infinite Squares

You are given **N** two-dimensional points $P_1, P_2, P_3, \ldots\ldots, P_N$. These points are **randomly generated one by one**. The coordinates of the points are non-negative and less than 10^5. It is possible that two different points have the exact same coordinates.

After the i-th point is generated, a square can be drawn in many different ways so that it contains all of the points that are generated till now. The side length and the orientation of the square can be anything. We consider that a square contains a point if and only if the point lies inside the square or on its boundary.

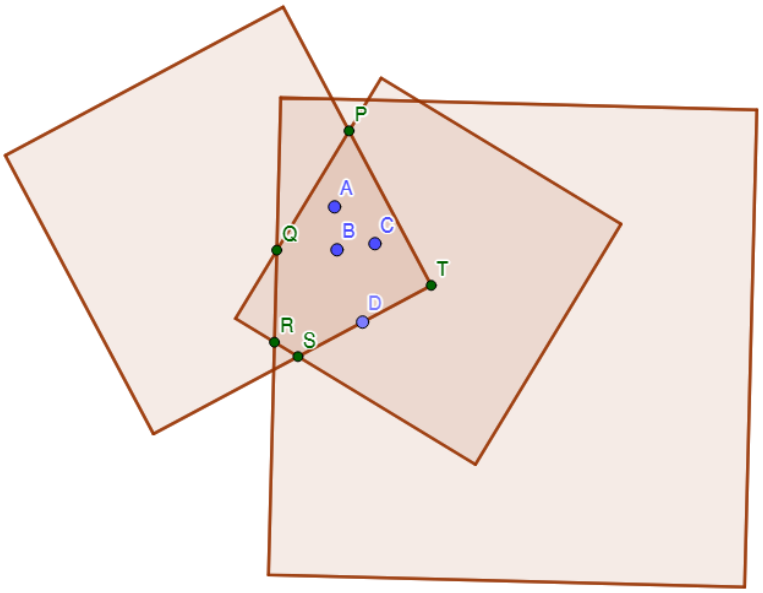
Let S_i be the set of all possible such squares which can individually contain all of the points from P_1 to P_i.

Let, the strength of a set **Q** containing some squares be:

Strength(Q) = 2 x (area of intersection of all squares in **Q**)

You need to determine the following sum:

$$\text{Sum} = \sum \text{Strength}(S_i) \quad (\text{where } 1 \le i \le N)$$



In the figure above, we want to cover the blue points (A, B, C, D) with any square. Each of the three squares drawn above contains all of the points individually. The strength of a set containing these three squares will be (2 x area of polygon PQRST). Note that for the given blue points, there are many squares other than these three, that can contain them individually.

Input

The first line will contain a single integer **T** denoting the number of test cases. Each test case will have an integer **N** and two integers **seed1** and **seed2**. You have to generate the points using **seed1** and **seed2** using the following pseudocode for random generator:

Random Generator Pseudocode:

```
global variables
    seed1, global var1
    seed2, global var2
end global variables

procedure GET-RANDOM()
  seed1 = (seed1 * 1103515243 + 12345) mod (2012345671)
  seed2 = (seed2 * 1092104927 + 54321) mod (2094828103)
  r = (seed1 * seed2) mod 100000
  return r
end procedure
```

Here you can find a sample C code for generating and printing **n** random points for **t** test cases. The following code is given so that you can better understand how to generate the **n** points using the seeds.

Sample C code for generating and printing n points :

```
#include<stdio.h>
long long int seed1, seed2;
int get_random() {
    seed1 = (seed1 * 1103515243 + 12345)%(2012345671);
    seed2 = (seed2 * 1092104927 + 54321)%(2094828103);
    int r = (seed1 * seed2) % 100000;
    return r;
}
int main(){
    int t, cs, i, n;
    scanf("%d",&t);
    for(cs = 1; cs<=t; cs++){
        scanf("%d %lld %lld",&n,&seed1,&seed2);
        for(i = 1; i<=n; i++){
            int x = get_random();
            int y = get_random();
            printf("%d %d\n",x,y);
        }
    }
    return 0;
}
```

Constraints

- **$1 \le T \le 10^3$**
- **$1 \le N \le 2 \times 10^7$**
- **$\sum N$ over all test cases $\le 2 \times 10^7$**
- **$1 \le seed1, seed2 \le 10^9$**

Output

For each case, print the case number and the value of **"Sum"** as described in the problem.

Please see the samples for more details.

Sample Input	Output for Sample Input
2 4 1 3 100000 842 91	Case 1: 5471244680 Case 2: 1997153783304545

Note About Sample

For the first test case in the sample, the 4 generated points are:

- (80412, 70783)
- (29920, 9845)
- (19576, 38405)
- (83584, 94275)

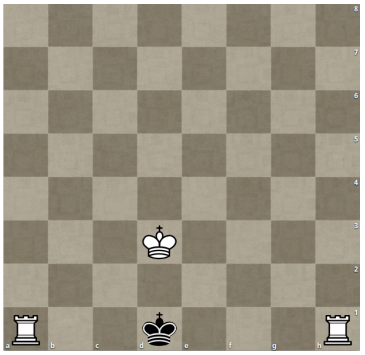
Problem B

Kings Vs Rooks

Chess is a two-player strategy board game played on a **8 x 8** board. Each player begins with **16** pieces where each type of piece moves differently. Player 1 has all the white pieces and player 2 all the black ones, with white being the one to move first and then players take turns alternatingly until the game ends. The game is played on a square board of eight rows and eight columns. The rows are called ranks and are numbered 1 to 8 from bottom to top from white's perspective. Similarly, the columns are called files and are denoted from a to h from left to right according to white's perspective. The objective of the game is to threaten the opponent's king in such a way that escape is not possible, i.e, checkmate the opponent. If a player's king is threatened, it is said to be in check and the player must remove the king from check on the next move by either moving to a safe square or capturing or blocking the attacking piece(s). The figure shows the starting position for the game of chess.

In this problem, we'll be considering a slightly modified version of chess with the following rules:

1. The game will be played on a **N x N** board
2. The game will consist of exactly 4 pieces - The white king, the black king, and two white rooks. All four pieces must be placed on different squares of the board.
3. Kings can move to any adjacent square (horizontally, vertically, or diagonally), unless the square is outside the board or occupied by a friendly piece or if the move would place the king under check.
4. Rooks can move to any square in the same rank or file, but not by jumping over other pieces. Also it cannot land on a square occupied by a friendly piece.
5. A piece **X** is threatened by another enemy piece **Y,** iff **Y** can reach **X** in exactly one legal move.
6. A piece **X** can capture another enemy piece **Y**, iff **X** can reach **Y** in exactly one legal move. On capturing, piece **Y** is removed from the board and **X** takes its place. Note that if **X** is the king, then it cannot capture **Y** if **Y** is guarded by an enemy piece. That is the king can never move and put itself in danger where it is attacked by an enemy piece.
7. The two kings cannot ever be in adjacent (horizontally, vertically or diagonally) squares.
8. The objective of the game is to place the four pieces on unique squares of the board in such a way that if it is black's move, then the black king is attacked and has no legal moves. That is, the black king must be threatened by at least one of the white rooks and has no way to move to a safe square or eliminate the checks (by capturing the attacking pieces if not guarded). Note that it is not necessary for the board configuration to be legal following the exact rules of chess. That is, if **N = 8,** there might be valid configurations in our modified game which might never appear in a game of chess. Like the figure shown beside, which can never occur in a proper game of chess given that it is black's turn to move, although it would be a perfectly valid configuration in our game.



Given **N**, you need to count how many ways you can place one white king, one black king, and two white rooks in the chessboard such that the black king is threatened and has no way to escape the threat given it is black's turn to move. Two configurations are different if either any of the squares contain different pieces or one square contains a piece but the other does not. Note that every square is uniquely represented by it's rank and file and hence mirrored or reversed configurations of the same configuration will be considered as different configurations. Also note that the white king, black king and white rooks are distinguishable but the two white rooks are identical. So in the figure above, if we swap the two rooks, the new configuration will be the same as the old one.

Since the number of ways can be huge, you need to output it modulo **7340033**.

Input

The first line contains the number of test cases denoted by **T (1 ≤ T ≤ 20)**. The next **T** lines contain the board dimension **N (1 ≤ N ≤ 100000)**.

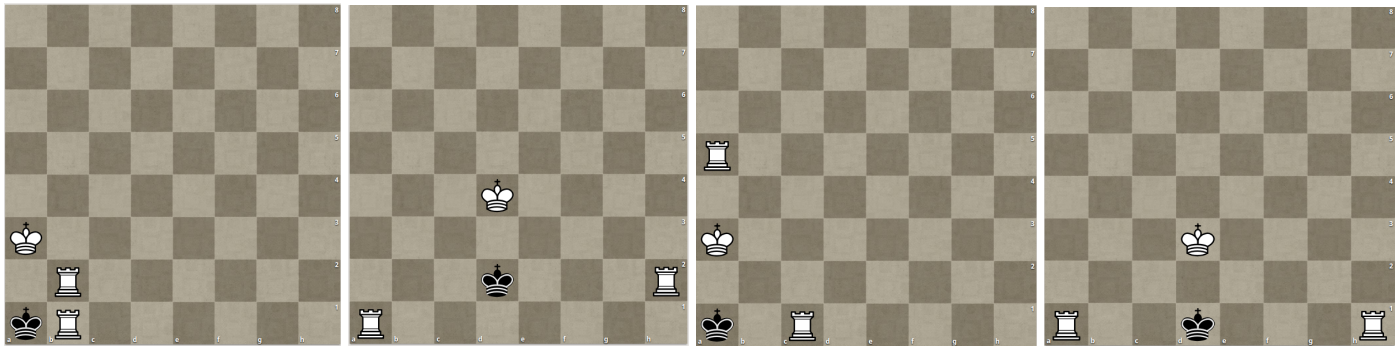
Output

For each case, output one line. First output the case number then the number of valid ways modulo **7340033**.

Sample Input	Output for Sample Input
8 1 2 3 4 5 6 7 8	Case 1: 0 Case 2: 0 Case 3: 232 Case 4: 1432 Case 5: 5188 Case 6: 14536 Case 7: 34464 Case 8: 72392

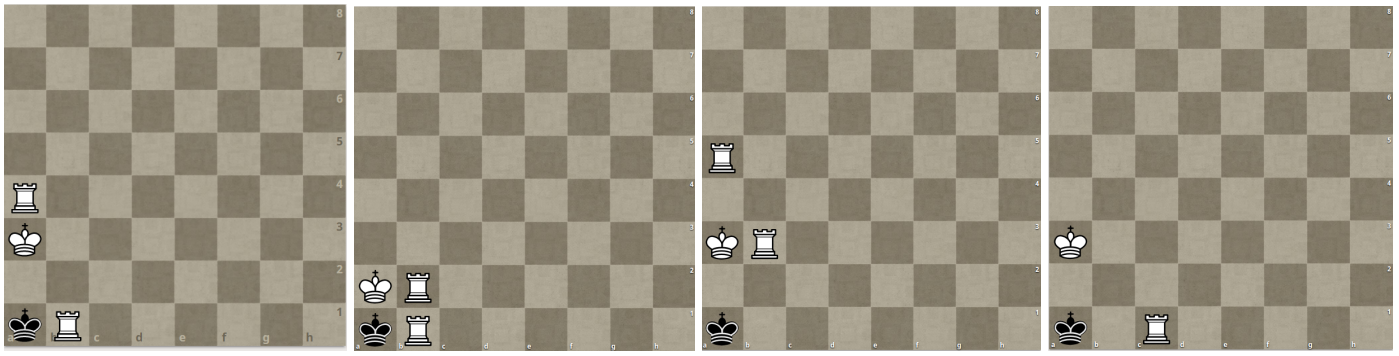
Explanation

For a **8 x 8** board, here are some valid configurations:



For all of these boards, the black king is attacked and has nowhere to go. Also both the white rooks are placed in the board and the kings are not in adjacent positions.

And here are some invalid ones:



For the first board, the black king can move and capture the attacking white rook thereby removing the check and moving to a safe square.

For the second board, although the black king is attacked and cannot capture anything because all the pieces are guarded, the two kings are adjacent which is not allowed. Hence it is not a valid checkmate position.

For the third board, it is stalemate. That is the black king has nowhere to go but it is also not attacked. So no checkmate.

For the fourth board, it is checkmate and the black king has nowhere to go. But we haven't placed both the rooks in the board and hence this is not a valid position.

Problem C

Almost A Palindrome?

You are given a string **S**, containing lowercase English letters only, and **Q** queries on it. Queries are of two types.

Query 1: (L, R)
You have to print the answer of this query in a single line. If the substring of **S** containing characters from **Lth** to **Rth** index (inclusive) is a palindrome, then the answer is 0. Otherwise you need to try to make this substring a palindrome, by removing exactly one character from it. If you can do so, then the answer for this query is the index of that character (1-based indexing) in **S**. If there are multiple such indices, then consider the smallest of them as the answer. If you cannot make the substring a palindrome by removing exactly one character, then the answer is -1.

Query 2: (X, C)
You have to set value **C** to **X-th** character in **S**.

P.S. You don't have to actually delete any characters from the string for any of the queries of type-1. This operation of removing an index is hypothetical. Queries of type-2 will persist for latter queries however.

Definitions
Palindrome: A string that reads the same backward as forward.
Substring: A contiguous sequence of characters within a string.

Input

First line of input will contain an integer **T (1 ≤ T ≤ 10)**, denoting the number of test cases. Each test case will start with a line containing a string **S**. Next line will contain an integer **Q**, denoting the number of queries to follow. Summation of **S** over all test cases, and summation of **Q** over all test cases will not exceed **200000**.

Each query will start with an integer **TYPE (1 ≤ TYPE ≤ 2)**. If **TYPE** is **1**, then it's a query of type-1, and it'll follow with two integers **L, R (1 ≤ L ≤ R ≤ |S|)**. Otherwise, it'll contain an integer **X (1 ≤ X ≤ |S|)** and, a character **C**, which is a lowercase English letter. This represents a query of type-2.

Output

For each test case, print the case number in a separate line. Then for each query of type-1, print the answer to that query according to the above definition, in a separate line. Refer to the sample I/O for more clarity on formatting.

Sample Input	Output for Sample Input
2 abca 5 1 1 4 1 2 4 2 4 w 1 1 4 1 3 3 xyz 4 1 1 2 1 1 3 2 3 x 1 1 3	Case 1: 2 -1 -1 0 Case 2: 1 -1 0

Explanation of Sample I/O

Explanation for Case-1:

Query-1: Removing character 'b' from the 2nd index will yield the substring "aca", which is a palindrome. Note that removing the 3rd index can create a palindrome as well. But we are looking for the smallest index, in case of multiple answers.

Query-2: There are no ways to make substring "bca" a palindrome.

Query-3: Now the string has become "abcw".

Query-4: Similar to query-2, there are no ways to make substring "abcw" a palindrome.

Query-5: It's already a palindrome.

Problem D

Winning Arrangements

N players are participating in a tournament. Before the tournament begins, the strength of each player is measured. In the tournament, the players are lined up in a certain order. **N-1** rounds are played. In each round any **two neighboring players** can be chosen and the stronger player wins. If both of them have the same strength then **either** can win. After the round the losing player leaves the tournament and the winning player goes back to their position. The players then fill the gap. The winner of the round also receives **1** bonus strength. A player wins the tournament if they are the last player remaining.

For each player **Pi**, find out how many initial arrangements of **N** players are there so that there is a way that player **Pi** could potentially win the tournament. Since the answer can be very large, print it modulo **1,000,000,007**.

Input

The first line of the input contains an integer T **(1 ≤ T ≤ 1,000)**, the number of test cases in the input file. Then **T** cases follow. Each test case starts with an integer **N (1 ≤ N ≤ 1,000)**, the number of players in the tournament. This is followed by a line with **N** integers corresponding to the strength of the players. Strength is a positive integer and can be at most **1,000,000,000**. It is guaranteed that ∑**N ≤ 50,000** over all cases in a single file.

Output

For each case, first print one line identifying the case number in the format **"Case X:"**. Then print **N** lines, the **i-th** line should correspond to the **i-th** player. Each line should contain an integer, the number of initial arrangements that could potentially lead to this player winning the tournament, modulo **1,000,000,007**.

Sample Input	Output for Sample Input
3 3 1 2 3 4 1 2 1 1 2 1 2	Case 1: 0 4 6 Case 2: 20 24 20 20 Case 3: 0 2

Explanation: In the second example, the first player won't be able to win in these 4 arrangements, [1, 2, 3, 4], [1, 2, 4, 3], [3, 4, 2, 1], [4, 3, 2, 1].

Note: In some slow languages (like python), it may not be possible to solve this problem.

Problem E

Alice, Bob and the Array

After a long break, we are having a programming contest in our country, and it's the preliminary contest for ICPC Dhaka Regional, 2020. Alice and Bob are here as visitors on this beautiful evening. They brought some colorful balloons for Dhaka as a token of appreciation. The host of the contest decided to keep those balloons in quarantine for next two weeks due to Covid19 issues. But, they did not want to go to the contest room empty handed and decided to present an array to the contestants.

So, Bob built an array with **N** integers within a very short time and wants to make it as beautiful as possible. He believes the beauty of an array is equal to its Maximum Consecutive Subarray Sum (MCSS) . MCSS of an array is the largest subarray sum of any contiguous non-empty subarray indexed from ***i*** to ***j*** such that $1 \leq i \leq j \leq N$.

Now, he asked Alice to perform a series of operations (if required) on the array to maximize its beauty. On each operation, she can select any two indices ***i*** and ***j*** such that $1 \leq i < j \leq N$ and reverse the subarray (***i*** and ***j*** inclusive).

As Alice is in a hurry she wants your help to do so. You have to maximize the beauty of the array performing the given operation as many times as it is required. If there are several ways to get the maximum beauty, you have to choose the one that requires the minimum number of moves.

Input

The first line of the input contains an integer **T (1≤T≤100)**, denoting the number of test cases. Each of the next T test cases has two lines of input. First line of each test case contains an integer **N (1≤N≤30000)** denoting the size of the array and the second line contains **N** space separated integers $a_1, a_2, a_3, \ldots a_N$ (**-100000**$\leq a_1, a_2, a_3, \ldots, a_N \leq$**100000**) representing the array. **Large dataset, use fast I/O methods.**

Output

For each test case print the test case number followed by the maximum consecutive subarray sum that can be achieved and the minimum number of moves. See the sample I/O section for output formatting.

Sample Input	Output for Sample Input
2 4 1 -10 2 2 2 1 2	Case 1: 5 1 Case 2: 3 0

Explanation

Case 1: Initial array is {1, -10, 2, 2}, One possible solution is as follows:
Consider i =1 and j =2 and reverse the subarray.
After performing the operation, the array will become {-10, 1, 2, 2}. So, Maximum consecutive subarray sum is 1 + 2 + 2 = 5.
Case 2: Initial array is {1, 2} and you do not need to perform any operation.

Problem F

Co Primes

Damien Rice is obsessed with songs, melodies and patterns. To create his next song, rootless tree, he has taken a keen interest in consecutive numbers. So he is trying to find a pattern in consecutive numbers but failed to do so, as he is a singer, not a mathematician or a programmer like you. But another member of his band, Lisa Hannigan, has thought about a problem involving two sequences of consecutive numbers starting from **a** and **b** respectively. Solution to this problem will help Damien to create the melody of this new song. For him, it takes a lot to solve this kind of problem. So, you have decided to help him. Now, it's your turn to fulfill one of your favorite faded fantasies of working with Damien Rice. Hurry up! Solve the following problem given by Lisa Hannigan.

Given **a**, **b** and **m**, find how many integers **i** (**0 ≤ i ≤ m)** exist such that **gcd(a + i, b + i) = 1.** Here, **gcd(x, y)** is the greatest common divisor of integers **x** and **y**.

Input

The first line will contain a single integer **T (1 ≤ T ≤ 1000)**. Each test case will contain three integers denoting **a**, **b** and **m**. Here, **1 ≤ a, b ≤ 10^{12} and 0 ≤ m ≤ 10^{12}.**

Output

For each test case, print the case number in a single line followed by the answer. Please see the sample for details.

Sample Input	Output for Sample Input
3 1 5 2 5 12 2 10 20 1	Case 1: 2 Case 2: 2 Case 3: 1

Explanation

For the first example, two pairs exist: (1, 5) and (3, 7).
For the second example, two pairs exist: (5,12) and (6, 13).
For the third example, only one pair exists: (11, 21).

Problem G

Strange Typewriter

You are given a strange typewriter, where each key of the typewriter contains a string consisting of lowercase alphabets. Consider these strings as **keyString**. You have to type a specific string **S** using this typewriter. If it is possible to type the string using this typewriter, you have to output the minimum number of keystrokes to write the string. Otherwise, you have to output "impossible".

Input

The first line of the input gives the number of cases, **T**. Each test case starts with a line containing the number of keys in the typewriter **N**. The following **N** line includes **N keyStrings** where keyString[i] corresponds to the i^{th} key. The following line will contain a single string **S** (containing lowercase alphabets) which you have to type.

Constraints:

- **1 ≤ T ≤ 10**
- **1 ≤ N ≤ 100000**
- **1 ≤** Summation of all |keyString| **≤ 1000000**
- **1 ≤ Summation of all |S| ≤ 1000000**
- **1 ≤ |Any string in input| ≤ 100000**

Output

For each test case, if it is possible to write, then print the minimum number of keystrokes required to write the string **S;** otherwise, print "impossible". See the sample outputs for the exact format.

Sample Input	Output for Sample Input
2 3 a b aab aab 2 a ba aab	Case 1: 1 Case 2: impossible

Problem H

Recursion, Lambda and Parameter

Mrs. Recursion runs a brokerage company. One day she came up with an idea of the custom fund offer. There are **N** types of stocks available to exchange in her brokerage. Each stock has two attributes. For the i'th stock they are: its price $\mathbf{P_i}$ and the average return $\mathbf{R_i}$. Now sometimes some of the stocks have such a high price that an individual may not afford it. So Mrs. Recursion came up with this revolutionary idea of the custom fund. Instead of buying a single stock, an individual may mix up the stocks. The price of this mixed up stock will have the weighted price. The average return of it will be the weighted average return of the constituent stocks. We call this mixed up stock- a fund. For example, someone may form one fund taking 0.5 stock from the first one, 0.3 from the second one and 0.2 from the third one (please note these fractions need to sum up to 1). Then the price of this fund is: $0.5P_1 + 0.3P_2 + 0.2P_3$ and the average return of this fund is $0.5R_1 + 0.3R_2 + 0.2R_3$. Needless to say, one can form a fund with one stock only, if they wish. For example one can form a fund just from the first stock and its price would be P_1 and the average return would be R_1.

Now here comes the Lambda with a brilliant consultancy idea. This is his advertisement:

You want to spend at most P money per fund?
I can tell you how to get the maximum average return.
You want to get at least R average return?
I can tell you what's the minimum fund price to achieve that.

However, Lambda does not know how to solve these problems. So he wants to hire a bunch of Parameters among you in this International Collegiate Parameters Competition.

Input

First line contains the positive integer **T**, the number of test cases. The first line of each test case starts with two positive integers **N**, the number of stocks and **Q**, the number of queries. There are **2N** non negative integers in the following line in the format of: $\mathbf{P_1\ R_1\ P_2\ R_2 \ldots P_N\ R_N}$**.** **Q** lines follow. Each line will be of one of the following formats:

1 P M: For at most **P** money per fund, what's the maximum average return one can achieve if one is allowed to mix at most **M** stocks to form the fund
2 R M: To achieve at least **R** average return, what would be the minimum fund price if one is allowed to mix at most **M** stocks to form the fund.

Constraints:

$1 \le T \le 10$
$1 \le N, Q \le 100{,}000$
$0 \le P_i, R_i, P, R \le 1{,}000{,}000{,}000$
$1 \le M \le N$.

Output

For each test case print the case number. Followed by the answers to the queries in different lines in the reduced fraction form X/Y (there is no common positive divisor of X and Y other than 1). Since X and/or Y may be too big, print them modulo 2^{64}, that is, you will actually print (X mod 2^{64})/(Y mod 2^{64}). Please note, if the answer is an integer, Y should be 1. It may be the case that the solution does not exist. In such a case print 0/0. For clarity check the sample input-output.

Sample Input	Output for Sample Input
3 2 3 5 10 5 20 1 4 2 2 5 2 2 15 2 2 3 5 5 10 5 1 6 1 1 6 2 2 10 1 2 1 10 10 20 20 1 15 2	Case 1: 0/0 5/1 5/1 Case 2: 5/1 5/1 0/0 Case 3: 15/1

Explanation

In the third case, there are two stocks. If we weight the stocks as 0.5, our fund price would be: 0.5 * 10 + 0.5 * 20 = 15 which yields 0.5 * 10 + 0.5 * 20 = 15 average return. It turns out this is the maximum average return with a fund price of 15.

Problem I

Lattice Triangle

In the Cartesian coordinate system a lattice point is a point whose abscissa and ordinate are both integers. For example (3, 4) is a lattice point, but (3.5, 4) or (√2, √3) are not lattice points. A lattice polygon is a polygon whose vertices are all lattice points. There is an unknown lattice triangle **TRI**. It has **L** lattice points inside (Points on the boundary are not considered inside). If the coordinates of all its vertices (both abscissa and ordinate) are multiplied by **n** (a rational number given in the form **p/q**, here **p** and **q** are positive integers) we get another lattice triangle (the value of n will be such that we will get another lattice polygon). The number of lattice points inside this new lattice triangle is $\mathbf{L_2}$. Given the value of **L,** $\mathbf{L_2}$ and n you have to find the area of **TRI**. You can assume that for the given values there will always be two such lattice triangles.

Input

There will be at most **20000** cases. Each line contains two positive integers which denotes the value of **L (0 ≤ L ≤ 100000)** and $\mathbf{L_2}$ **(0 ≤** $\mathbf{L_2}$ **≤** $\mathbf{10^{13}}$**)** and a fraction in the form **p/q** (Here **p** and **q** are positive integers and **p ≤ 10000** and **q ≤ 500** and **p** and **q** are not equal). Input is terminated by a line where all integer values are zero. This line should not be processed.

Output

For each line of input produce one line of output. This line should contain an integer that denotes twice the area of **TRI**.

Sample Input	Output for Sample Input
0 3 2/1 1 4 2/1 0 0 0/0	4 3

Problem J

COVID - 19

COVID - 19 has been the most prominent topic in our lives for the last year. Just when some of us thought that the pandemic was under our control, we are going through another huge increase in infection rate. There is still very little data to predict what is going to happen with this virus. So as general people we should really listen to the experts. We should wear a mask whenever we are interacting with another person. We should avoid unnecessary movements completely. We should sanitize our hands frequently. Most importantly, we should keep our distances. Recommended is six feet. This distance keeping among humans is being called Social Distancing.

Let's say there is a corridor with length N feet. The width of the corridor is really narrow. For simplification we can imagine the corridor as a line of length N. Let's also assume that each human can be imagined as a point in this corridor. What is the maximum number of humans that can be in the corridor maintaining Social Distancing? For example if N = 6, then we can have two humans on two opposite ends of the corridor. If N = 12, then we can put three humans safely, two on the end points, and one in the middle of the corridor. But if N = 5, then we have no way to put more than one human in the corridor. Because we won't be able to maintain six feet distance between them.

Input

Input will have a single integer **N ($1 \leq N \leq 10^4$)**, denoting the size of the corridor in feet.

Output

Print the maximum number of people you can put in the corridor maintaining Social Distancing.

Sample Input 1	Output for Sample Input 1
5	1

Sample Input 2	Output for Sample Input 2
6	2

Sample Input 3	Output for Sample Input 3
12	3

Sample Input 4	Output for Sample Input 4
13	3