Information About Persons

The secret service of Landia decided to manage information about the whole people of Landia. Unfortunately, its computer scientists are completely helpless. The input looks as follows. The first line contains the number of records, $n \leq 200\,000$. On the next n lines, every record stores information about one person:

- personal code (a 31-bit positive integer);
- status (two to seven nonspace characters); the first character is always `M` or `F` (sex); if the second character is `-` (minus), the person is dead; otherwise, if one of the next characters is `C`, the person is computer scientist; if one character is `S`, the person works for the secret service;
- name of the person (8 to 35 characters, including space).

Why are the computer scientists of the secret service so helpless? The above list contains a lot of duplicates: every time a person changes its status or name, the new record is simply added at the end of the list.

The secret service wants to know:

- how many men are living in Landia;
- how many women are living in Landia;
- how many computer scientists are working in Landia;
- how many persons are working for the secret service;
- the list of computer scientists who work for the secret service (in alphabetical order); display this list after one empty line.

Beware: consider only the last (actual) status of every person.

Sample Input

```
12
10 FAB Blue Mary
15 MCA Sonk Jean
10 FAS Rode Mary
20 FXS Aske Vera
20 F- Aske Vera
12 FDE Kant Mia
12 FSE Kant Mia
16 MUV Barbe Louis
21 MSC Conta Pietro
24 FCS Ando Maria
16 MCS Barbe Louis
25 FET Donk Laura
```

Output for Sample Input

```
3 men
4 women
4 computer scientists
5 for secret service

24 FCS Ando Maria
16 MCS Barbe Louis
21 MSC Conta Pietro
```

B

Wedding

One year after winning the Sapientia ECN contest, exactly on the date of ECN 2018, Andrei is getting married! Andrei is usually late. Maybe this is acceptable for different meetings, perhaps even those with his bachelor's thesis coordinator, but surely it is absolutely unacceptable to be late at your own wedding!! However, Andrei woke up late today as well, after all night he was up analyzing the stock market.

The area between Andrei's house and the wedding place is represented by a matrix of size $N \times M$. Andrei's house is at cell (1, 1), and the wedding takes place at cell (N, M). Each minute Andrei can move to any of the 8 neighbor cells; that differ from his current cell with +1 / 0 / –1 on the line or column number.

There are also K slot machines around that Andrei is addicted to. Each slot machine i, with $1 \le i \le K$; is defined by three numbers: $S_i.x$, $S_i.y$, the line and column number of its position; and $S_i.a$ the *addictiveness* of slot machine S_i. For each free cell in the matrix, the *danger* level is defined by the sum of the addictiveness of each slot machine, proportional to the distance to it. More precisely, for a free cell at position (X, Y), the danger is:

$$danger(X,Y) = \sum_{i=1}^{K} \left(S_i.a \times \frac{N+M}{|S_i.x - X| + |S_i.y - Y|} \right)$$

Andrei needs to get to his wedding in at most T minutes. Knowing his addiction, he wants to choose a path for which the maximum *danger* level is as low as possible, while reaching his bride Lidia on time.

Input Specification

The first line of the input contains four numbers: N, M, K, and T, the size of the matrix, the number of slot machines, and the time Andrei has in minutes. Each of the following T lines contains three numbers: $S_i.x$, $S_i.y$, and $S_i.a$, the coordinates and addictiveness of slot machine i.

Input Limits, Constraints, and Clarifications

- $1 \le N, M \le 1000$;
- $1 \le K \le 50$;
- $1 \le S_i.a \le 100$;
- $1 \le S_i.x \le N$;
- $1 \le S_i.y \le M$;

- $\max(N, M) < K \leq N \times M$.
- Andrei's path cannot contain a cell with a slot machine.
- There are no slot machines at (1, 1) or (N, M).
- There is always a solution: love will find a way!
- Congratulations and best wishes for Lidia and Andrei!

Output Specification

Output one number, the minimum value of the danger he has to assume to reach the wedding on time. Print this value with exactly three decimals.

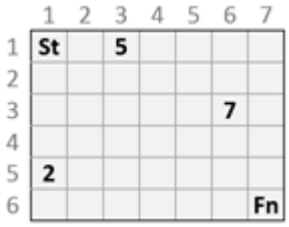
Sample Input

```
6 7 3 7
1 3 5
3 6 7
5 1 2
```

Output for Sample Input

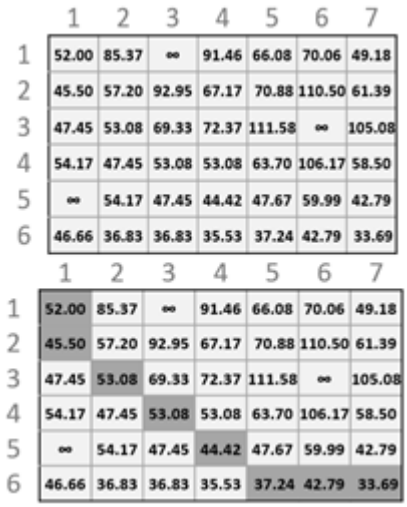
```
53.080
```

Explanation



The matrix is shown on the figure above. Andrei is at **St**, the wedding is at **Fn**, and the danger of the three slot machines is written at their positions.

The danger of each cell is shown in the top matrix of the following figure:



The danger of cell (2, 3) is:

$5 \times 13 / (|2 - 1| + |3 - 3|) + 7 \times 13 / (|2 - 3| + |3 - 6|) + 2 \times 13 / (|2 - 5| + |3 - 1|) =$
$5 \times 13 / 1 + 7 \times 13 / 4 + 2 \times 13 / 4 = 65 + 22.75 + 5.2 =$ **92.95**

Andrei can reach the wedding in 7 minutes, on a path with 8 cells total, out of which the most dangerous are the cells with danger **53.08**. All paths with a smaller maximum danger would be longer, and Andrei would be late.

C

Invasion

Our world contains many powerful things, but everyone agrees that nothing is of greater power than a war helicopter. Especially the newest model equipped with a flamethrower that can fire in 4 directions simultaneously.

Thus, when the aliens invaded the world, there was no question about what weapon should be used against them.

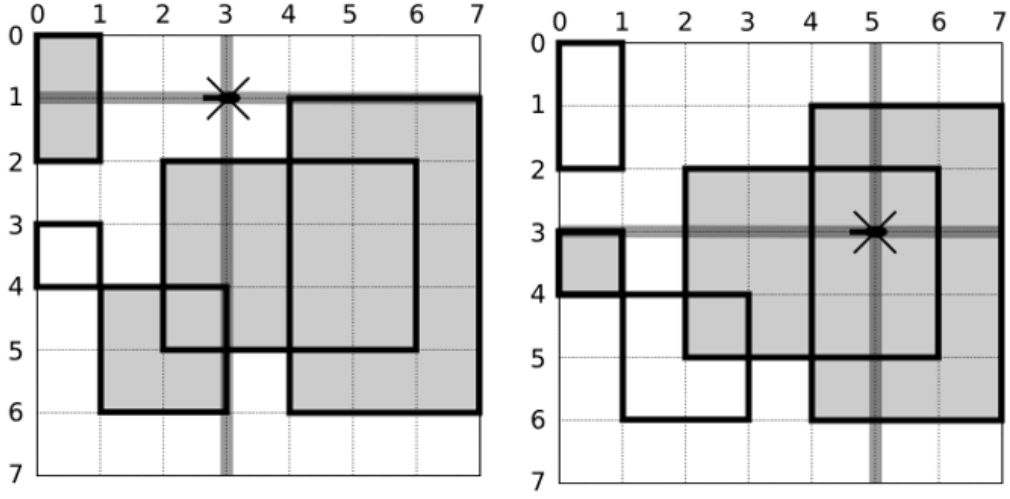
The invaders have *N* spaceships, each of which can be represented as a grid-aligned rectangle. These rectangles can overlap. There are *M* potential target points, where the helicopter can be placed. When a target point is selected, the helicopter flies there and fires its weapon, throwing destructive fire in all four directions (left, right, up, down). These fire beams don't stop until they reach the border of the world and destroy each enemy spaceship that happens to be in their way.

Your job is to write a program that determines for each target point how many spaceships would be destroyed if the helicopter were to shoot its weapon there.

All coordinates (corner points of rectangles, target points) are integers (both *X* and *Y*).

If only the border or a corner point of a spaceship's rectangle is hit, it should still be counted. If two or more overlapping spaceships are hit, all of them should be counted. If a spaceship is hit by multiple beams, it should be counted only once. If the helicopter is inside the rectangle of a spaceship, that spaceship should be counted.

For example:



On the left figure, the helicopter is at (3,1), and 4 buildings get destroyed.

On the right figure, the helicopter is at (5,3), and 3 buildings get destroyed.

Input Specification

The first line contains N (the number of spaceships) and M (the number of target points) separated by a space. Each of the following N lines contains four space-separated integers: L, R, T, and B; which are the left, right, top, and bottom coordinates of a spaceship's rectangle ($L < R$; $T < B$). Each of the following M lines contains two integers separated by a space: X and Y, the coordinates of a target point.

Input Limits and Constraints

- $1 \leq N \leq 100\,000$;
- $1 \leq M \leq 100\,000$;
- each coordinate is an integer between 0 and 999 999 999 (both included).

Output Specification

The output should contain M lines: for each target point, the number of spaceships destroyed should be printed if that target point is selected.

Sample Input

```
5 2
0 1 0 2
0 1 3 4
1 3 4 6
2 6 2 5
4 7 1 6
3 1
5 3
```

Output for Sample Input

```
4
3
```

D

Run a FRACTRAN program

One of the most brilliant mathematicians of all time, J. H. Conway at Princeton University, discovered a “Simple Universal Programming Language for Arithmetic”, which he named FRACTRAN.

The idea is deceptively simple but unexpectedly powerful and is based on the next *fraction game*. There is given a finite list of fractions

$$f_1, f_2, \ldots, f_k$$

and a starting integer n. “You repeatedly multiply the integer you have at any stage (initially n) by the earliest f_i in the list for which the product is integral. Whenever there is no such fit, the game stops.” (Quotation from T. M. Cover et al. (eds.), *Open Problems in Communication and Computation*, Springer-Verlag New York Inc., 1987) Thats all.

Here it is the program, i.e. the “game”, which lists... all the prime numbers!

$$\frac{17}{91}, \frac{78}{85}, \frac{19}{51}, \frac{23}{38}, \frac{29}{33}, \frac{77}{29}, \frac{95}{23}, \frac{77}{19}, \frac{1}{17}, \frac{11}{13}, \frac{13}{11}, \frac{15}{2}, \frac{1}{7}, \frac{55}{1}$$

More exactly, if you start with $n = 2^2$, then whenever a power of two appears as the value of n, then that is exactly the subsequent prime power of 2:

$$2^3, 2^5, 2^7, 2^{11}, 2^{13}, 2^{17}, \ldots$$

Question: How many steps does it take starting with $n = 2^2$ to obtain $n = 2^{37}$?

Input Specification

There is no input for this problem.

Output Specification

The output should contain a single integer, the number of steps required.

E

Raising the Glasses

Team Red and Team Blue play a drinking game, where teams alternate turns. Team Red chooses the number of glasses N from an interval $[a..b]$ and also plays the first turn.

After N is chosen, glasses numbered from 1 to N are lined up orderly in a single line with glass #1 at the leftmost and glass #N at the rightmost.

In each turn, the captain of the team next in line, selects exactly one glass among the existing ones. The selected glass and its immediate adjacent glasses to the left and right (if any) are picked up and taken away by the team. The team who cannot make any move loses the game. In other words, the team who makes the last move and takes away the last glass or glasses wins.

Observe that there are values for N for which the victory of Team Red can be guaranteed. As in these cases there exists a sure-win strategy for Team Red, we call these winning positions.

For example, $N = 1$, $N = 2$, or $N = 3$ are such winning positions, as Team Red removes all glasses in one move, winning the game. For $N = 4$, there's no sure-win strategy, whatever Team Red moves first, will result in one or two adjacent glasses remaining on the table, which will be cleared by Team Blue in the next turn. $N = 5$ is again a winning position for Team Red, the winning strategy consists in picking first glass #3. This will leave 2 non-adjacent glasses, namely glass #1 and glass #5 on the table. Only one of them can be cleared by Team Blue in the next turn, leaving one behind for Team Red to secure the victory.

Your job is to help Team Red. Given the lower bound a and the upper bound b for the number of glasses, you have to determine the smallest number of glasses N ($a \le N \le b$) that is winning position for Team Red, or return –1 if no such winning position exists in the $[a..b]$ interval.

Input Specification

The first line of the input contains an integer T ($T \le 10\,000$) denoting the number of cases. Each case contains two integers a and b ($1 \le a \le b \le 1\,000\,000\,000$) in a line, denoting the lower bound and upper bound from which the smallest winning position N must be chosen if it exist.

Output Specification

For each test case, output in a line the smallest integer *N* that is winning position for the respective input, or –1 if there is no sure-win strategy.

Sample Input

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

Output for Sample Input

```
1
2
3
-1
5
```

F

Circular Matrix Matching

A $p \times r$ matrix $B = (B_{ij})_{0 \le i < p,\ 0 \le j < r}$ occurs in $m \times n$ matrix $A = (A_{ij})_{0 \le i < m,\ 0 \le j < n}$ starting at position (k, l) if $B_{ij} = A_{k+i,\ l+j}$ for $0 \le i < p$, $0 \le j < r$, and m, n, p, $r > 0$, where the indices are considered circularly. For example, let:

$$B = \begin{pmatrix} a & b & c \\ g & a & b \end{pmatrix}, \qquad A = \begin{pmatrix} a & b & \underline{a} & b & c & f & g \\ g & a & g & a & b & a & b \\ c & a & b & 4 & 5 & \underline{a} & b \\ b & c & d & e & f & g & \underline{a} \end{pmatrix}$$

B occurs in A beginning at positions: (0, 2), (2, 5), and (3, 6) (positions of underlined elements in the matrix).

For given B and A, find all starting positions at which B occurs in A.

Input Specification

The first line of the input contains the number of test cases. For each test case, the first line contains the number of rows p and the number of columns r for matrix B, separated by a space. Each of the next p lines contains the elements of a row of matrix B, each element being a character, separated by a space. The next line contains the number of rows m and the number of columns n for matrix A, separated by a space. Each of the next n lines contains the elements of a row of matrix A, each element being a character, separated by a space.

Input Limits and Constraints

- $1 \le p \le 100$, $1 \le r \le 100$, $1 \le m \le 100$, $1 \le n \le 100$;
- each matrix element is a character.

Output Specification

For each test case, print in a line #*i*, where *i* is the number of the test case, followed by a line containing the found positions in lexicographic order in the form (*k*,*l*), separated by a space. If there is no solution, leave a blank line.

Sample Input

```
2
2 3
a b c
g a b
4 7
a b a b c f g
g a g a b a b
c a b 4 5 a b
b c d e f g a
2 2
a a
a a
5 5
a a a a a
a a a a a
a a a a a
a a a a a
a a a a b
```

Output for Sample Input

```
#1
(0,2) (2,5) (3,6)
#2
(0,0) (0,1) (0,2) (0,3) (0,4) (1,0) (1,1) (1,2) (1,3) (1,4) (2,0) (2,1) (2,2) (2,3) (2,4) (3,0) (3,1) (3,2) (4,0) (4,1) (4,2)
```

G

Smart City

Imagine a smart city where (1) only self-driving cars are used, (2) all streets are bidirectional, cars pass each of them in one time unit, and, consequently, the road network of the city can properly be modelled by a connected, unweighted, undirected graph.

Given such a graph (n: number of nodes; m: number of edges; nodes are denoted by 1, 2, ..., n), for each pair of nodes A and B, we imagine having one unit of "flow" along the edges from A to B. More precisely, the flow between A and B divides itself evenly along all the possible *shortest* paths from A to B: so if there are k shortest paths from A to B, then $1/k$ units of flow pass along each one.

We define the traffic of an edge to be the total amount of flow it carries, counting flow between all pairs of nodes using this edge.

We are interested in the most vital edge of the graph, the one that carries the highest volume of traffic along shortest paths.

Input Specification

The first line of the input contains the number of test cases. For each test case, the first line contains two integers, the values of n and m (separated by a single space). The next m lines contain the endpoints of the edges (separated by a single space).

Input Limits and Constraints

- $1 < n < 1000$;
- $0 < m < 20\,000$.

Output Specification

For each test case, print one line with three integers separated by a single space: the highest traffic value and the endpoints of the corresponding edge. If there are multiple edges with maximal traffic value, then print the first one in lexicographical order.

Sample Input

```
2
2 1
1 2
14 17
1 2
1 3
2 3
3 7
4 5
4 6
5 6
6 7
7 8
8 9
8 12
9 10
9 11
10 11
12 13
12 14
13 14
```

Output for Sample Input

```
1 1 2
49 7 8
```

H

Server

You are in charge of a server that needs to run some submitted tasks on a first-come, first-served basis. Each day, you can dedicate the server to run these tasks for at most T minutes. Given the time each task takes, you want to know how many of them will be finished today.

Consider the following example. Assume $T = 180$ and the tasks take 45, 30, 55, 20, 80, and 20 minutes (in the order they are submitted). Then, only four tasks can be completed. The first four tasks can be completed because they take 150 minutes, but not the first five, because they take 230 minutes which is greater than 180. Notice that although there is enough time to perform the sixth task (which takes 20 minutes) after completing the fourth task, you cannot do that, because the fifth task is not done yet.

Input Specification

The input consists of a single test case. The first line contains two integers n and T, where $1 \leq n \leq 50$ is the number of tasks, and $1 \leq T \leq 500$. The next line contains n positive integers not greater than 100, indicating how long each task takes, in the order they are submitted.

Output Specification

Display the number of tasks that can be completed in T minutes on a first-come, first-served basis.

Sample Input 1

```
6 180
45 30 55 20 80 20
```

Output for Sample Input 1

```
4
```

Sample Input 2

```
10 60
20 7 10 8 10 27 2 3 10 5
```

Output for Sample Input 2

```
5
```

I

Addition of Points

It is given a curve $E : y^2 = x^3 + a \cdot x$ defined over a finite field $F_p = \mathbb{Z}_p^*$ where p is a prime number, and an extra point $\mathcal{O}$, that lives "at infinity".

Your task is to determine the value of t, for which $P = \underbrace{P + \cdots + P}_{\text{t additions on E}} = t \cdot Q$, where P, Q two points on E. It is known that $t \in \{0, 1, \ldots n-1\}$ where n is the smallest integer such that $n \cdot P = \mathcal{O} \in F_p$, so n is the order of the point P.

Below is given the addition algorithm for two points on such a curve, where p is a prime number grater than 3.

1. If $P_1 = \mathcal{O}$, then $P_1 + P_2 = P_2$.
2. Otherwise, if $P_2 = \mathcal{O}$, then $P_1 + P_2 = P_1$.
3. Otherwise, write $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$.
4. If $x_1 = x_2$ and $y_1 = -y_2$, then $P_1 + P_2 = \mathcal{O}$.
5. Otherwise, define λ by

$$\lambda = \begin{cases} \dfrac{(y_2 - y_1)}{(x_2 - x_1)}, if\ P \neq Q \\ \dfrac{(3 \cdot x_1^2 + a)}{2 \cdot y_1}, if\ P = Q \end{cases}$$

and let $x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda \cdot (x_1 - x_3) - y_1$. Then $P_1 + P_2 = (x_3, y_3)$.

Write a program that for several value of (p, a, n, P, Q) calculates the values of t. The input contains several test cases. The first line shows the number of test cases followed by a blank line. There will be also a blank line between every two consecutive test cases.

Each test case consist of one line: the first three values show p, A, n in this order, followed by the value of P and Q each of them put in brackets and separated by a comma.

For each test cases try to find the corresponding t value. After each value of t you must put a blank line. For each test case it is true that the value of t is less than n. The value of p consist in max 32 bit.

Sample Input

```
3

229 1 239 (5, 116) (155, 166)

23 1 14 (6, 4) (13, 16)

743059 1 742919 (67975, 118275) (467929, 594620)
```

Output for Sample Input

```
176

12

20019
```

Speed Pizza

A local pizza restaurant offers pizza delivery services to customers directly to their home. The restaurant receives orders and delivers the pizza using motor scooters. Orders are taken in the order of their arrival and assigned to a driver. Each driver will have his own scooter and can only pick up a single pizza order for delivery to a location. After each delivery, a driver returns to the restaurant to pick up the next order, if any available, until all orders are delivered. The location of an order can be any location on the map, but will never be the location of the restaurant. Locations are linked by two-way roads which have an associated cost, the time needed for a motor scooter to go from a location to another.

The manager of the restaurant wants to deliver the orders received using the available scooters as soon as possible in order to keep the customers happy. You are given:

- a map of locations numbered from 1 to *l*, linked by two-way roads with a time cost;
- the location of the restaurant;
- the total number of available scooters;
- the orders the restaurant receives.

Your task is to find the minimum cost to deliver all orders to their location. The orders must be taken in the order they arrive.

Input Specification

The first line of the input consists of a pair of numbers *l* and *r*, representing the number of locations (numbered from 1 to *l*) and the number of roads which link them. On the next *r* lines, the two-way roads are listed, given in the form of location *l*1, location *l*2, and cost *c*, meaning that from location *l*1 to location *l*2 (and vice versa) the time to arrive is *c*.

On the next line, the location *p* of the restaurant is given.

After the restaurant location, two numbers representing the number of scooters *s* and the number of orders *o* are provided on a new line. The final line contains the locations for the *o* orders received by the restaurant, separated by a space.

Input Limits and Constraints

- $1 < l < 100\,000$;
- $1 < s < o$;
- the order locations will always be accessible from the restaurant;
- the time cost of delivering an order will never be higher than 1 000 000 000;
- there is no limitation on the number of scooters on a road at a moment in time;
- all the orders are considered delivered when all the scooters have returned to the restaurant, and no order is to be taken.

Output Specification

The output should contain a single line representing the minimum cost to deliver all the orders to their destination locations.

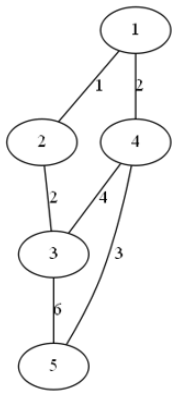
Sample Input

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

Output for Sample Input

```
12
```

Explanation



Restaurant: 1

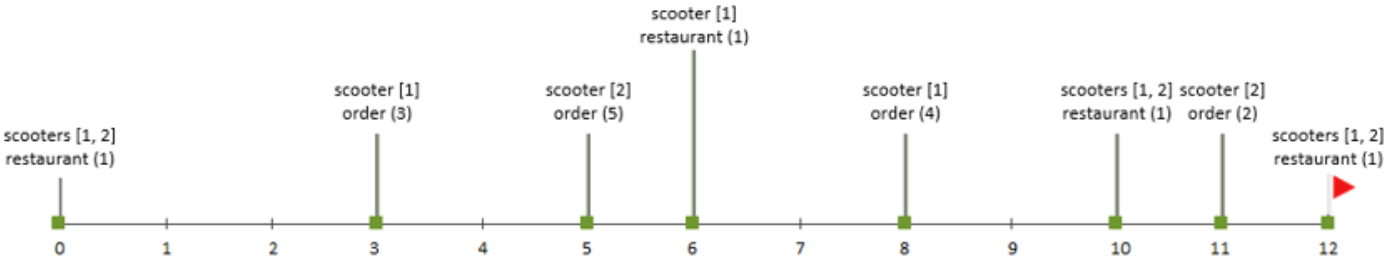
Number of scooters: 2

Order locations: 3 5 4 2

The computation for the minimum-cost route and the order-scooter allocations are displayed in the table below:

Order location	Minimum route to location	Minimum cost (roundtrip)	Scooter 1	Scooter 2
3	1 → 2 → 3	6	x	
5	1 → 4 → 5	10		x
4	1 → 4	4	x	
2	1 → 2	2		x
Total time			10	12

The order timeline is shown below:



Note: In the example above, the final order (2) could have been taken by any of the two scooters, as at time 10 they were both at the restaurant.

K

p_order

Let a be a list that stores a permutation $a = (a_0, a_1, a_2, ..., a_{n-1})$ of order n, and let p be a natural number. We say that the permutation is p-sorted if for any $x > 0$ index, it is true that $a_x > a_{[(x-1)/p]}$, where $[(x-1)/p]$ is used to denote the integer part of $(x-1)/p$.

For example, (1,3,2,5,4) is a 3-sorted order 5 permutation, because $a_1 = 3 > 1 = a_0$, $a_2 = 2 > 1 = a_0$, $a_3 = 5 > 1 = a_0$, $a_4 = 4 > 3 = a_1$.

There exist altogether 12 different order 5 permutations that are 3-sorted. These are the following: (1,2,3,4,5), (1,2,3,5,4), (1,2,4,3,5), (1,2,4,5,3), (1,2,5,3,4), (1,2,5,4,3), (1,3,2,4,5), (1,3,2,5,4), (1,3,4,2,5), (1,3,5,2,4), (1,4,2,3,5), (1,4,3,2,5).

Given n and p, calculate the number of p-sorted order n permutations **modulo 666013**.

Input Specification

The input has 10 lines, each of which containing natural numbers n and p, separated by a space ($1 < p < n \leq 600\,000$). Each tuple represents a problem that needs to be solved.

Output Specification

The output needs to contain 10 numbers in separate lines, giving the corresponding answers for the 10 input problems.

Sample Input

```
5 2
5 3
6 3
1234 8
5000 100
141414 200
414141 101
345 234
54321 321
600000 81
```

Output for Sample Input

```
8
12
40
530546
141922
142664
471427
526978
151098
596797
```