1. There are 6 problems, 0 pages, in this programming contest.
2. The input data is stored in a file, and it may contain many test cases. Your program must process all the test cases in the file.
3. For reading and writting files, use *standard input* and *standard output* in your program such as `stdin` and `stdout` in **C**.

Problem A
Jacobi Symbol
Time limit: 1 seconds

Let m and n be two positive integers, and n is odd. Define a function, denoted by $\left(\frac{m}{n}\right)$, recursively as follows. Note that, in the following definition, $a \equiv b \pmod{n}$ means that a and b have the same remainder after they are divided by n. For example, $3 \equiv 10 \pmod{7}$.

1. If $\gcd(m, n) > 1$ then $\left(\frac{m}{n}\right) = 0$. Otherwise, apply the following rules.
2. If n is odd, then $\left(\frac{2}{n}\right) = \begin{cases} 1 & \text{if } n \equiv \pm 1 \bmod 8, \\ -1 & \text{if } n \equiv \pm 3 \bmod 8. \end{cases}$
3. If n is odd, and $m_1 \equiv m_2 \pmod{n}$, then $\left(\frac{m_1}{n}\right) = \left(\frac{m_2}{n}\right)$.
4. If n is odd, then $\left(\frac{m_1 m_2}{n}\right) = \left(\frac{m_1}{n}\right)\left(\frac{m_2}{n}\right)$
   In particular, $\left(\frac{2^k t}{n}\right) = \left(\frac{2}{n}\right)^k \left(\frac{t}{n}\right)$
5. If m and n are both odd, then $\left(\frac{m}{n}\right) = \begin{cases} -\left(\frac{n}{m}\right) & \text{if } m \equiv n \equiv 3 \bmod 4, \\ \left(\frac{n}{m}\right) & \text{otherwise.} \end{cases}$

In the above definition, $-a \bmod n$ is equivalent to $n - a$. Thus, $\pm 1 \bmod 8$ is equivalent to 1 or 7 mod 8. Write a program to compute $\left(\frac{m}{n}\right)$ efficiently.

Input Format

Each test case contains two integers m and n in one line. The value of m and n are both less than 2^{31}, and n is an odd integer, $n > 2$.

The test data file may contain many test cases. The last test case is followed by a line containing a single 0.

Output Format

The outputs for each test case m and n is the value of the function $\left(\frac{m}{n}\right)$. Print it in one line.

Sample Input

```
3 3
3 5
2 7
0
```

Sample Output for the Sample Input

```
0
-1
1
```

Problem B
Number of sRobots
Time limit: 2 seconds

NCPC Company is interested in investing in cheap sRobots for its assembly plant. The assembly plant is rectangular in shape and the floor can be seen as having MxN virtual grids. Due to carelessness of the human workers, assembly parts are often dropped to the floor, and remain there until they are picked up by sRobots. At end of a work day, a team of sRobots is scheduled to enter the assembly plant to pick-up those loose parts. The sRobots are queued near grid $(1, 1)$ of the assembly plant. sRobots must enter at grid $(1, 1)$ and exit at grid (M, N) of the assembly plant. To prevent interference, only one sRobot can enter and remain in the assembly plant at any given time. When a sRobot is inside the assembly plant, say at grid (i, j), it repeatly performs the following two actions until it exits the assembly plant at (M, N).

1. If there are loose parts on the floor at grid (i, j), picks-up all the loose parts at grid (i, j).
2. Moves to either grid $(i + 1, j)$ or grid $(i, j + 1)$.

As these are cheap robots, pre-scan of the assembly plant for route planning is not possible. Please write a program to determine the minimum number of sRobots that must enter the assembly plant in order to clean up all the loose parts on the floor.

Table 1: A 7x8 plant with 10 grids with loose parts (x) on the floor

7x8	1	2	3	4	5	6	7	8
1								
2						x		
3		x						x
4			x		x			
5		x			x			
6								x
7					x	x		

Input Format

A test case of the problem consists of several lines of input. The first line contains 3 integers, M, N, and P, denoting that the assembly plant has size MxN, and there are P grids with loose parts on the floor. The next P lines each contains two integers x, y, denoting that grid (x, y) has loose parts on the floor. Note that $1 \leq M, N, P \leq 10,000$. There may be more than one test caes. The last test case is followed by a line containing three 0s.

Output Format

The outputs for each test case is a single integers on a line, indicating the minimum number of sRobots that must enter the assembly plant in order to clean up all the loose parts on the floor.

Sample Input

```
1 1 1
1 1
7 8 10
1 5
2 1
2 7
3 2
3 4
4 1
4 4
5 7
6 4
6 5
0 0 0
```

Sample Output for the Sample Input

```
1
3
```

Problem C
Switching Game
Time limit: 1 seconds

Consider a line of n switches, from left to right, either all initially in the OFF position or all initially ON. In order to avoid turning the switches by accident, the i-th switch can be flipped only when the $(i-1)$-th switch is ON and all the switches on the left of the $(i-1)$-th switch are OFF. The first switch can always be flipped. There are two goals depending on the initial state of the switches. If the initial state has all the switches OFF, then we want to have the n-th switch ON and the rest OFF. If the initial state has all the switches ON, then we want to have all the switches OFF. Each time only one switch can be flipped.

For example, consider the followings:

```
Example 1:

   [OFF, OFF, OFF]
1. [ON , OFF, OFF]
2. [ON , ON , OFF]
3. [OFF, ON , OFF]
4. [OFF, ON , ON ]
5. [ON , ON , ON ]
6. [ON , OFF, ON ]
7. [OFF, OFF, ON ]


Example 2:

   [ON , ON , ON ]
1. [OFF, ON , ON ]
2. [OFF, ON , OFF]
3. [ON , ON , OFF]
4. [ON , OFF, OFF]
5. [OFF, OFF, OFF]
```

Write a program to count the number of times the n switches must be flipped to achieve the goals. Since the number can be very large, output the answer modulo with 1,000,000,009.

Input Format

The first line of the input gives the number of test cases, T (≤ 50). T test cases follow. Each test case consists of one line that has two integers n ($3 \leq n \leq 100000$) and s (s =0 or 1), where n indicates the number of switches and s indicates the initial state of switches. If $s = 0$, then it indicates the initial state has all the switches OFF; else $s = 1$ indicates the initial state has all the switches ON.

Output Format

For each test case, output your answer modulo 1,000,000,009.

Sample Input

```
3
3 0
100 0
3 1
```

Sample Output for the Sample Input

```
7
336286277
5
```

Problem D
Heap Detection
Time limit: 3 seconds

The max-heap data structure can be represented as a nearly complete binary tree. The max-heap property is that for every node x other than the root the value in node x is smaller than or equal to the value in its parent. A nearly complete binary tree can be stored in an array as follows: Each node of the tree corresponds to an element of the array. The values in the tree can be stored from the root level by level into an array. The nodes in the same level are stored from left to right. You are asked to determine whether an array is a max-heap.

Input Format

In each problem instance, there is a line containing the values in an array. Note that there is a blank between any two integers in each line. The number of elements in an array is at most 100. They are positive integers in the range $[1, 10000]$.

Note also that the test data file may contain more than one instance. The last instance is followed by a line containing a single 0.

Output Format

For each test case, output 1 in a line if it is a max-heap; otherwise, output 0 in a line.

Sample Input

```
16 14 10 4 7 9 3 2 8 1
16 14 10 8 7 9 3 2 4 1
0
```

Sample Output for the Sample Input

```
0
1
```

Problem E
Constrained Seat Arrangement
Time limit: 1 seconds

There are n^2 seats organized in a 2-dimensional $n \times n$ square x-y room. The seat coordinates range from 0 to $n-1$ in both x and y dimensions. There are n persons to be arranged to sit in these n^2 seats. The seat arrangement rules are as follows.

1. There are no two persons sitting on seats in the same row or in the same column. (the row placement constraint and the column placement constraint)
2. Two seats with coordinates (x_1, y_1) and (x_2, y_2) where $|x_1 - x_2| = d$ and $|y_1 - y_2| = d$ are said to be of diagonal direction and have diagonal distance d. Any two selected seats of diagonal direction should have diagonal distance d greater than a given diagonal distance limit k (i.e. $d > k$). (the diagonal placement constraint)
3. A forbidden strip of seats may be specified with i, $x_{i,start}$, and $x_{i,end}$ that corresponds a horizontal strip of seats between $(x_{i,start}, i)$ and $(x_{i,end}, i)$ inclusively in row i. Zero or more forbidden strips can be specified for persons to be placed in any row. Thus, a person cannot be arranged to sit on seats in any forbidden strip(s). (the forbidden strip constraint)

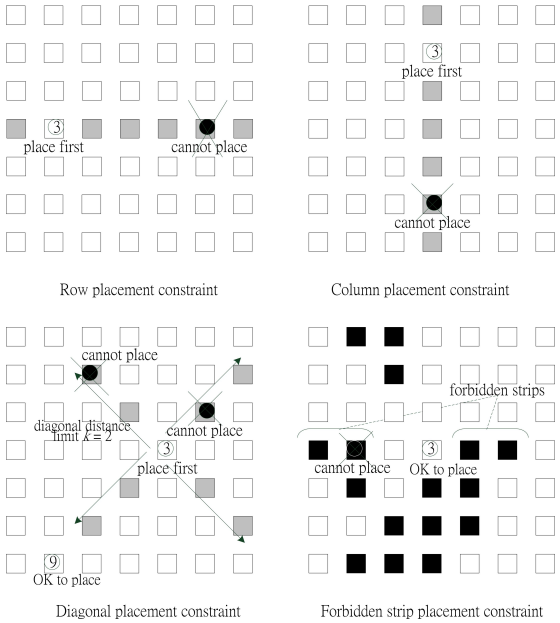


Figure 1: Four placement constraints of room layout.

Given t sets of seat arrangement test data, your program is to iteratively calculate and output the number of all satisfied seat arrangements for each test data set in the order of given test data sets.

Input Format

The first line consists of an integer t representing the number of test cases. It is followed by t sets of test data. Each test data set contain two lines. In each test data set j, the first line consists of three integers n_j, k_j, and c_j representing the size of each dimension (the number of persons), the diagonal distance limit, and the number of forbidden strip constraints, respectively. The second line in each test data set contains the set of forbidden strip constraint with three integers i_j, $x_{i,start,j}$, and $x_{i,end,j}$ representing row number, starting column number,and ending column number. After listing all test data sets, the last line contains a single 0.

Output Format

For each test case set j, output the number m_j of all satisfied seat arrangements in one line.

Sample Input

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

Sample Output

```
1
3
4
```

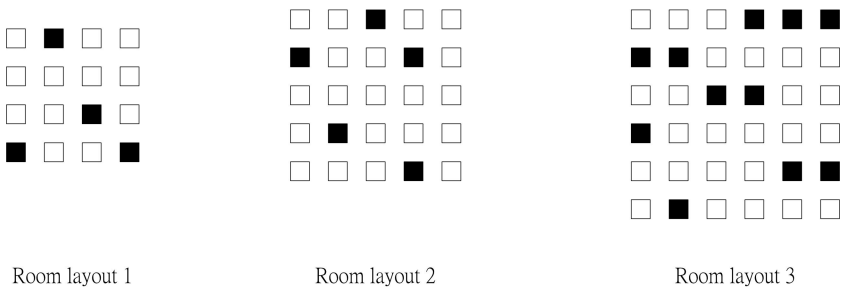
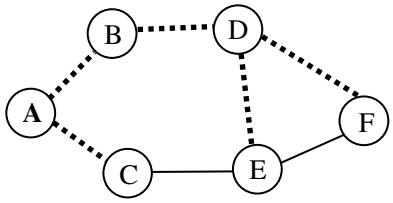


Figure 2: Illustration of sample input with marked forbidden strips.

Problem F
Aggregation Trees
Time limit: 1 seconds

A wireless sensor network is a multi-hop network in which sensor nodes can be randomly deployed and the data transmission between nodes usually involves other intermediate nodes. Wireless sensor networks usually connect to the outside world through the so called sink (may be regarded as a gateway). All data collected by sensor nodes are sent to the sink hop by hop, and then the sink relays such data to remote users or servers through the Internet, satellite, or any viable medium. Given a sensor network with the unique sink and a set of sensor nodes, an aggregation tree is often constructed to collect the desired sensing information. The root of the tree is the unique sink, and the tree should connect every sensor in the network. For example, an aggregation tree (marked in dash lines) is built in the following graph where A is the sink. Evidently, an aggregation tree is also a spanning tree.

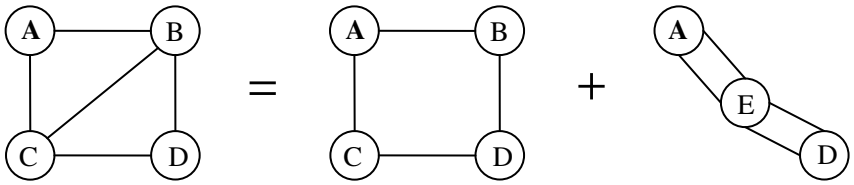


Since sensor nodes are usually equipped with scarce battery power and are thus limited in their active lifetime, how to select an aggregation tree becomes a major issue in wireless sensor networks. In order to find out the best aggregation tree, a brute-force method is enumerating all possible different aggregation trees for further processing. Now we consider a simpler problem for computing the number of different aggregation trees firstly. The following concept could be useful to tackle the above problem. Note that the following graphs (in the middle and on the right) each have four spanning trees, and the left graph has eight (8=4+4) spanning trees. Also note that the middle graph is the resulting graph by deleting edge (B, C) from the left graph, and the right graph is the resulting graph by deleting edge (B, C) from the left graph and identifying them into vertex E. You can see that the number of spanning trees of the left graph is the sum of that of the middle graph and the right graph.

Please write a program to find out the number of different aggregation trees for a given sensor network.

Input Format

There are at most 10 test cases. The first line of each instance consists of an integer n ($2 \leq n \leq 30$), where n is the number of nodes (vertex set of G) labeled with 1, 2, ..., n in the network.



The second line of each instance is the label of the unique sink in $\{1, 2, \ldots, n\}$. The third line of each instance is m ($1 \leq m \leq 30$) the number of communication links in the network. It is then followed by m pairs of integers (separated by a space), which indicate communication links (edges in E). The last test case will be followed by a line containing $Z = 0$.

Output Format

The output for each instance should contain an integer denoting the number of different aggregation trees of the given network.

Sample Input

```
4
1
5
1 2
1 3
2 3
2 4
3 4
0
```

Sample Output for the Sample Input

```
8
```