# 2021 年中国大学生程序设计竞赛全国邀请赛 (湖南)
# 暨第十二届湘潭市大学生程序设计竞赛

**June 6, 2021**

## A. A+B Problem

HZ has a special calculator, in which all the integers are 11-bit signed integers. The range of an 11-bit signed integer is from $-2^{10}$ to $2^{10}-1$ (from $-1024$ to $1023$). When using this calculator to calculate the sum of two integers $A$ and $B$, the result may be different from other calculators. Here are some examples:

$$
\begin{aligned}
1+1 &= 2\\
1023+1 &= -1024\\
1023+2 &= -1023\\
-1024+(-1) &= 1023\\
-1024+(-2) &= 1022
\end{aligned}
$$

HZ found this special calculator very strange, so he comes to you. You are given two integers $A$ and $B$, and you need to guess the result of $A+B$ of this calculator.

### Input

The first line of input contains an integer $T$ ($1 \le T \le 10^5$), denoting the number of test cases.

Each test case contains two integers $A$ and $B$ in one line. $-1024 \le A, B \le 1023$.

### Output

For each test case, print one integer in one line, denoting your answer.

### Sample

| Input | Output |
|---|---|
| 5<br>1 1<br>1023 1<br>1023 2<br>-1024 -1<br>-1024 -2 | 2<br>-1024<br>-1023<br>1023<br>1022 |

# B. Binary Number

You are given two binary numbers $x$ and $y$ ($0 \leq y \leq x$). $x$ is variable while $y$ is constant. You need to perform some steps of operations on $x$ and finally make $x$ equal to $y$.

In each step, you can perform one of the two operations below.

1. Make $x$ become $x - 1$. This operation can be performed only if $x > 0$.
2. If the $i$-th most significant bit of $x$ is 1, and the $i + 1$-th most significant bit of $x$ is 0, we can swap the two bits in $x$. That is to say, for two adjacent bits, if the "left" is 1 and the "right" is 0, we can swap them (change 10 to 01).

Now you need to find the minimum steps we need to make $x$ become $y$.

## Input

The first line of input contains an integer $T$, denoting the number of test cases.

For each test case, there is a line containing two binary integers $x$ and $y$.

The total length of all binary integers in a test file is not larger than 1000000.

## Output

For each test case, output an integer in a line, denoting the minimum steps to make $x$ become $y$.

## Sample

| Input | Output |
|---|---|
| 5<br>10100 1000<br>1000001 11111<br>10010 0<br>11111 101<br>101001 10010 | 4<br>3<br>7<br>11<br>4 |

# C. Calculate

You are given 4 positive integers $x_1, x_2, y_1, y_2$. Now you need to calculate

$$\sum_{i=x_1}^{x_2}\sum_{j=y_1}^{y_2}\left(\left\lfloor\frac{i}{x_1}\right\rfloor+\left\lfloor\frac{x_2}{i}\right\rfloor+\left\lfloor\frac{j}{y_1}\right\rfloor+\left\lfloor\frac{y_2}{j}\right\rfloor\right)^2$$

where $\lfloor x\rfloor$ denotes the biggest integer that is not bigger than $x$.

The answer may be too large, so you just need to output it modulo $(10^9+7)$.

## Input

The first line of input contains an integer $T$ $(1 \le T \le 100)$, denoting the number of test cases.

Each test case contains 4 positive integers $x_1, x_2, y_1, y_2$ in one line. $1 \le x_1 \le x_2 \le 10^9$. $1 \le y_1 \le y_2 \le 10^9$.

## Output

For each test case, print one integer in one line, denoting your answer modulo $(10^9+7)$.

## Sample

| Input | Output |
|---|---|
| 5<br>1 1 1 1<br>1 2 3 4<br>2 5 2 5<br>1 12345678 2 23456789<br>3 123456789 4 1000000000 | 16<br>100<br>490<br>841328555<br>80052041 |

## D. Car

HZ is driving a car along a straight road of length $L$. Now the car is at one end of the road, and HZ wants to go to the other end.

There are $N$ speed measuring points on the road. The distance between the start point and the $i$-th speed measuring point is $S_i$. The speed limit of the $i$-th speed measuring point is $V_i$, which means that when the car passes the point, its velocity should not exceed $V_i$.

The initial velocity of the car is zero, and the car must **stop** at the other end. The absolute value of acceleration of the car can not exceed $A$.

HZ wants you to calculate the shortest time to go to the other end of the road, but Zayin thinks it's too easy. Therefore, they come up with new problems for you.

Initially, there is no speed measuring point on the road. Then, the $N$ speed measuring points will be built one by one in order from 1 to $N$. After each one is built, you need to answer whether the shortest time becomes longer or not.

### Input

The first line of input contains an integer $T$, denoting the number of test cases.

For each test case, the first line contains 3 integers $L$, $A$ and $N$ $(1 \le L \le 10^5,\ 1 \le A \le 9,\ 1 \le N \le 10^5)$.

In the next $n$ lines, each line contains 2 integers $S_i$ and $V_i$ $(0 \le S_i \le L,\ 1 \le V_i \le 10^3)$, denoting a speed measuring point. And they will be built in order of input.

It's guaranteed that the sum of $N$ of all the $T$ test cases is not larger than $10^5$.

### Output

For each test case, print $N$ lines.

In the $i$-th line of each test case, if the shortest time becomes longer after building the $i$-th one, print 1. Otherwise, print 0.

### Sample

| Input | Output |
|---|---|
| 1<br>6 2 5<br>2 1<br>4 1<br>3 3<br>1 1<br>1 1 | 1<br>1<br>0<br>1<br>0 |

# E. CCPC Strings

A string whose each character is either “C” or “P” is called a *CP-String*, so there are $2^N$ different *CP-Strings* of length $N$. We define the *CCPCness* of a *CP-String* as the number of the most non-overlapping “CCPC” as its substrings.

For example, the *CCPCness* of “CCPCCCPC” is 2, but the *CCPCness* of “CCPCCPC” is 1 because the chosen “CCPC” substrings can not overlaps.

You are given an integer $N$, and you need to calculate the sum of *CCPCness* of all the $2^N$ different *CP-Strings* of length $N$. The answer may be too large, so you just need to output it modulo $(10^9 + 7)$.

## Input

The first line of input contains an integer $T$ $(1 \leq T \leq 10^5)$, denoting the number of test cases.

For each test case, there is one line with one integer $N$ $(1 \leq N \leq 10^9)$.

## Output

For each test case, print one integer in one line, denoting your answer modulo $(10^9 + 7)$.

## Sample

| Input | Output |
|---|---|
| 7<br>3<br>4<br>5<br>6<br>7<br>10<br>123456789 | 0<br>1<br>4<br>12<br>31<br>417<br>11557040 |

# F. Control in a Matrix

You are given an $N \times M$ matrix $A[1..N][1..M]$. We say that $(x_1, y_1)$ controls $(x_2, y_2)$ if and only if $A[x_1][y_1] > A[x_2][y_2] + |x_1 - x_2| + |y_1 - y_2|$. Now you need to calculate the number of pairs $((x_1, y_1), (x_2, y_2))$ that $(x_1, y_1)$ controls $(x_2, y_2)$.

## Input

The first line of input contains an integer $T$, denoting the number of test cases.

For each test case, the first line contains two integers $N$ and $M$, denoting the size of the matrix.

Then there are $N$ lines denoting the matrix, where each line contains $M$ integers.

$1 \le N, M \le 10^3,\ 1 \le A[i][j] \le N + M$. And the sum of $N \times M$ of all $T$ test cases is not larger than $10^6$.

## Output

For each test case, print an integer in one line, denoting your answer.

## Sample

| Input | Output |
|---|---|
| 2<br>3 3<br>1 1 1<br>6 6 6<br>1 1 1<br>3 3<br>1 2 3<br>4 5 6<br>6 6 6 | 18<br>14 |

## G. Game

Alice and Bob are playing a game. They take turns and Alice moves first. There is a set of positive integers. In one's turn, he or she can choose a number (suppose $x$) in the set, and choose a positive integer $k$ ($k$ does not need to be in the set), and replace $x$ with $x - (10^k - 1)$. For example, you can make a number $x$ in the set become $x - 9$ or $x - 99$ or $x - 999$ or $x - 999\cdots$. After that the set must still be a set of positive integers. That is to say:

1. The number must still be positive: $x - (10^k - 1) > 0$.
2. A set can not have duplicate elements: $x - (10^k - 1)$ can not be equal to any other numbers in the set.

They take turns and Alice moves first. The one who can't move loses the game. Now the question is that who will win the game if they play optimally.

### Input

The first line of input contains an integer $T$, denoting the number of test cases.

For each test case, the first line contains a number $N$, denoting the number of numbers in the set.

The second line contains $N$ different positive integers $A_1, A_2, \ldots, A_N$, denoting the numbers in the set.

It's guaranteed that $1 \le A_i \le 10^9$, and the sum of $N$ of all $T$ test cases is not larger than $10^5$.

### Output

For each test case, print "A" for Alice or "B" for Bob in one line, denoting the winner.

### Sample

| Input | Output |
|---|---|
| 5<br>3<br>1 2 3<br>3<br>2 11 20<br>3<br>11 12 13<br>3<br>10 19 28<br>2<br>100 1000 | B<br>B<br>A<br>A<br>A |

# H. Huge Directed Graph

There is a huge directed graph which contains $10^{18}$ nodes numbered from 1 to $10^{18}$. There is a directed edge from $x$ to $y$ if and only if $x < y \leq 500x$, and the length of the edge is $\ln \left\lfloor \left\lfloor \sqrt{\frac{y}{x}} \right\rfloor^{\frac{3}{2}} \right\rfloor$, where ln is natural logarithm, and $\lfloor x \rfloor$ denotes the biggest integer that is not bigger than $x$.

You are given two integers $x$ and $y$ ($x < y$), and you need to find the **longest** path from $x$ to $y$. If the longest path is $d$, you just need to output $\lfloor e^d \rfloor$, where $e$ is the base of natural logarithm.

## Input

The first line of input contains an integer $T$ ($1 \leq T \leq 200000$), denoting the number of test cases.

In the next $T$ lines, each line contains two integers $x$ and $y$ ($1 \leq x < y \leq 10^{18}$).

## Output

For each test case, print one integer in one line, denoting $\lfloor e^d \rfloor$.

## Sample

| Input | Output |
|---|---|
| 4<br>2 8<br>3 27<br>4 64<br>1 12345678987654321 | 2<br>5<br>8<br>1163817123840 |

## Explanation

For the last sample, one of the longest paths is:

$1 \to 81 \to 6569 \to 1478082 \to 378389069 \to 96867601845 \to 27994736933456 \to 12345678987654321$

# I. Sequence

We define the uniqueness of a sequence as the number of unique numbers in the sequence. For example, the uniqueness of $\{1, 2, 1, 2\}$ is 0 because there is no unique number, and the uniqueness of $\{5, 6, 7, 6, 6\}$ is 2 because 5 and 7 are unique numbers.

You are given a sequence with length $N$. You need to cut it into $M$ parts (each part is a continuous subsequence), and maximize the sum of the uniqueness of the $M$ parts.

## Input

The first line of input contains an integer $T$, denoting the number of test cases.

For each test case, the first line contains 2 integers $N$ and $M$.

The second line contains $N$ integers $A_1, A_2, \ldots, A_N$, denoting the sequence.

$1 \le M, A_i \le N,\ 2 \le M \le 10$. The sum of $N$ of all the $T$ test cases is no more than $10^5$.

## Output

For each test case, print one integer in one line, denoting the maximal sum of the uniqueness.

## Sample

| Input | Output |
|---|---|
| 4<br>2 2<br>1 1<br>3 2<br>1 1 1<br>4 2<br>1 2 2 1<br>6 3<br>1 1 2 2 3 3 | 2<br>1<br>4<br>4 |

# J. Stacks

There are $N$ stacks, numbered from 1 to $N$. Initially, the $i$-th stack only contains a number $i$. Now there are $M$ move operations, and the $i$-th operation is to move all the numbers in stack $a_i$ to stack $b_i$. Specifically speaking, we pop all the numbers in stack $a_i$ one by one, and push them to stack $b_i$ in the order of popping (the first to be popped is the first to be pushed).

After all the $M$ operations, you should output the numbers in the stacks.

## Input

The first line of input contains an integer $T$, denoting the number of test cases.

For each test case, the first line contains 2 integers $N$ and $M$ $(1 \le M \le N \le 10^5)$.

In the next $M$ lines, each line contains 2 integers $a_i$ and $b_i$ $(1 \le a, b \le N,\ a \ne b)$, denoting an operation.

It's guaranteed that the sum of $N$ of all the $T$ test cases is no more than $10^5$.

## Output

For each test case, print $n$ lines. The $i$-th line begins with an integer $S_i$, denoting the number of numbers in the $i$-th stack, followed by $S_i$ integers, denoting the numbers in the $i$-th stack from top to bottom.

## Sample

| Input | Output |
|---|---|
| 1 | 0 |
| 6 5 | 1 6 |
| 1 2 | 0 |
| 2 3 | 0 |
| 3 4 | 5 4 2 1 3 5 |
| 4 5 | 0 |
| 6 2 | |

# K. Substring

You are given a string $S[1..N]$ containing only lowercase letters. Now you need to find the longest substring $S[l..r]$ such that every letter (“a” to “z”) appears no more than $K$ times in the substring. You just need to output the length $(r - l + 1)$ of the longest substring.

## Input

The first line of input contains an integer $T$, denoting the number of test cases.

Each test case contains one integer $K$ $(1 \leq K \leq N)$ and one string $S$ in one line.

It’s guaranteed that the sum of lengths of the input strings is no more than $2 \times 10^5$.

## Output

For each test case, print one integer in one line, denoting the length of the longest substring.

## Sample

| Input | Output |
|---|---|
| 3<br>1 abcabcabc<br>2 abcabcabc<br>2 aaabbbccc | 3<br>6<br>4 |

# L. Swap

You are given a sequence of length $N$, and there is also a number $H$ in your hands.

In each step, you can swap the number in your hands with one of the numbers in the sequence.

Now you need to find the minimum steps to make the sequence in non-decreasing order.

## Input

The first line of input contains an integer $T$, denoting the number of test cases.

For each test case, the first line contains two integers $N$ and $H$, denoting the length of the sequence and the number in your hands initially.

The second line contains $N$ integers $A_1, A_2, \ldots, A_N$, denoting the initial sequence.

$1 \le H, A_i \le 10^9$. And the sum of $N$ of all $T$ test cases is not larger than 100000.

## Output

For each test case, print an integer in one line, denoting your answer.

## Sample

| Input | Output |
|---|---|
| 4<br>4 3<br>1 4 2 5<br>4 3<br>1 2 5 4<br>6 2<br>2 3 4 2 3 4<br>6 2<br>3 4 5 3 4 5 | 2<br>1<br>3<br>4 |