Colombian Collegiate Programming League
CCPL 2018

Round 5 – April 28

Problems

This set contains 11 problems; pages 1 to 12.

(Borrowed from several sources online.)

Official site `http://programmingleague.org`

Follow us on Twitter `@CCPL2003`

A - Arrange Some Marbles

Source file name: `arrange.c`, `arrange.cpp`, `arrange.java`, *or* `arrange.py`

You are given some marbles of n different colors. You have to arrange them in a line. The marbles adjacent with same color form a group. In each group there can be 1 to 3 marbles. Any two adjacent groups should have different colors and sizes. The first and last groups also should have different colors and sizes. You are given the number of each of these n marbles. You have to count the number of ways you can arrange them in a line maintaining the above constraints.

For example if you have 4 red marbles and 4 green marbles, then you can arrange them in the following 8 ways:

```
GGGRRGRR   GGRGGRRR   GGRRRGGR   GRRGGGRR
RGGRRRGG   RRGGGRRG   RRGRRGGG   RRRGGRGG
```

Input

Input contains multiple number of test cases. The first line contain the number of test cases t ($t < 3000$). Each test case starts with n ($1 \leq n \leq 4$), the number of different colors, followed by n integers. The i-th integer denotes the number of marbles of color i. The number of marbles of any color is within the range 0..7 (inclusive). The color of the marbles are numbered from 1 to n.

The input must be read from standard input.

Output

For each test case output contains one integer in one line denoting the number of ways you can arrange the marbles.

The output must be written to standard output.

Sample Input	Sample Output

6
2 3 3
2 4 4
2 6 6
3 3 4 5
3 4 5 6
4 2 3 4 5
	0
	8
	12
	174
	1234
	1440

B - Be Efficient

Source file name: `be.c`, `be.cpp`, `be.java`, *or* `be.py`

Consider an integer sequence consisting of N elements, where:

$$X_0 = A$$
$$X_i = ((X_{i-1} \cdot B + C) \bmod M) + 1 \qquad \text{for } i \in \{1, \ldots, N-1\}$$

You will be given the values of A, B, C, M, and N. Find out the number of consecutive subsequences whose sum is a multiple of M.

Consider an example where $A = 2$, $B = 1$, $C = 2$, $M = 4$, and $N = 4$. So, $X_0 = 2$, $X_1 = 1$, $X_2 = 4$, and $X_3 = 3$. The consecutive subsequences are {2}, {2 1}, {2 1 4}, {2 1 4 3}, {1}, {1 4}, {1 4 3}, {4}, {4 3}, and {3}. Of these 10 'consecutive subsequences', only two of them add up to a figure that is a multiple of 4: {1 4 3} and {4}.

Input

The first line of input is an integer T ($T \geq 0$) that indicates the number of test cases. Eact case consists of 5 integers A, B, C, M, and N. A, B, and C will be non-negative integers not greater than 1000. N and M will be a positive integers not greater than 10 000.

The input must be read from standard input.

Output

For each case, output the case number followed by the result.

The output must be written to standard output.

Sample Input	Sample Output

2
2 1 2 4 4
923 278 195 8685 793
	Case 1: 2
	Case 2: 34

C - Candy Chain

Source file name: `candy.c`, `candy.cpp`, `candy.java`, *or* `candy.py`

A Candy Chain is a sequence of individual candies. Candies come in 26 different flavors identified by the lowercase letters *a* to *z*. Margot has a particularly fancy Candy Chain displayed in her shop: *axsa*.

After school, children will come to her and buy portions of the Candy Chain. Every child has different preferences. For example there is a child who likes portions with flavors *ababi*, and is willing to pay 3 euros for it. Another child likes portions with flavors *axsa* and is willing to pay 5 euros for it.

Margot can extract portions of the Candy Chain and sell them to the children. When she extracts a portion, she joins the remaining left and right parts of the Candy Chain, and can then continue serving additional portions, or decide to stop.

The same sequence of flavors can be sold multiple times to the same child (as long as Margot is able to extract multiple instances of it from her Candy Chain). Margot never throws away candies if she cannot sell them. She can reverse candy portions while selling them (e.g., *axsa* and *asxa* are equivalent). Margot does not have to serve all children and she does not have to serve the children in any particular order.

Your task is to help Margot compute the maximum amount she can earn from her Candy Chain.

Input

The input file contains several test cases. The first line of each test case represents Margot's Candy Chain as a non-empty string of characters, at most 50 characters. The second line consists of the number of children, an integer $1 \leq C \leq 200$. The following C lines represent the preferences of each child as two items separated by a space: his or her preferred candy portion, represented by a non-empty string of characters; and what he or she is willing to pay, represented by an integer $0 \leq P_i \leq 1\,000\,000$, for $1 \leq i \leq C$. You can assume that all strings are formed of the lowercase characters *a* to *z* only.

The input must be read from standard input.

Output

For each test case, output the maximum amount Margot can earn.

The output must be written to standard output.

Sample Input	Sample Output

`acmicpcxxxacmzacmzacmzmca`
`5`
`icpc 5`
`cpci 1`
`acm 2`
`acmacm 10`
`xxx 0`
	`21`

D - Zeroes III

Source file name: `zeroes.c`, `zeroes.cpp`, `zeroes.java`, *or* `zeroes.py`

The factorial $n!$ of a natural number n is defined as

$$n! = 1 \cdot 2 \cdot 3 \cdot \cdots \cdot n$$

The function $F_1(n)$ as

$$F_1(n) = 1! \cdot 2! \cdot 3! \cdot \cdots \cdot n!$$

And the function $F_2(n)$ as

$$F_2(n) = F_1(1) \cdot F_1(2) \cdot F_1(3) \cdot \cdots \cdot F_1(n)$$

Given two natural numbers n and b, your job is to find the number of trailing zeroes in $F_2(n)$ when expressed in base b.

Input

The input contains several test cases. Each line contains two integers n ($1 \leq n \leq 1\,000\,000$) and b ($2 \leq b \leq 10\,000$). Input is terminated by a line containing two zeroes.

The input must be read from standard input.

Output

For each line of input produce one line of output with the number of trailing zeroes in $F_2(n)$, when expressed in base b.

The output must be written to standard output.

Sample Input	Sample Output

`10 3`
`4 2`
`0 0`
	`57`
	`8`

E - Eigensequence

Source file name: `eigen.c`, `eigen.cpp`, `eigen.java`, *or* `eigen.py`

Given an increasing sequence of integers $a_1, a_2, a_3, \ldots, a_k$, the E-transform produces a sequence $b_1, b_2, b_3, \ldots, b_k$ of the same length such that

- $b_1 = a_1$
- for $j > 1$, b_j is the only integer $a_{j-1} < b_l \leq a_j$, which is divisible by $a_j - a_{j-1}$.

For example, from $S = 0, 1, 4, 9, 16, 25, 36, 49$ one gets $E(S) = 0, 1, 3, 5, 14, 18, 33, 39$. A sequence S such that $E(S) = S$ is called an *eigensequence*. For instance, $S = 2, 3, 4, 6, 8, 12, 16, 18, 20$ is an eigensequence.

Given integers a_1 and a_n, how many eigensequences (of any length) start with a_1 and end with a_n?

Input

The input has many data lines, followed by a terminating line. Each line has two integers, a_1 and a_n. If $a_1 < n$, it's a data line. Otherwise it's a terminating line that should not be processed. On each line, $0 \leq a_1 \leq a_n \leq 44$. This guarantees that each output fits into a 32 bit integer.

The input must be read from standard input.

Output

For each data line, print a line with a_1, a_n, and x, where x is the number of eigensequences (of any length) that start with a_1 and end with a_n.

The output must be written to standard output.

Sample Input	Sample Output

`0 3`
`5 7`
`2 8`
`0 0`
	`0 3 3`
	`5 7 1`
	`2 8 12`

F - Triple-free Binary Tree

Source file name: `free.c`, `free.cpp`, `free.java`, *or* `free.py`

A binary string consists of ones and zeros. Given a binary string T, if there is no binary string S such that SSS (three copies of S concatenated together) is a substring of T, we say T is triple-free.

A pattern consists of ones, zeros, and asterisks, where an asterisk ($*$) can be replaced by either one or zero. For example, the pattern 0**1 contains strings 0001, 0011, 0101, 0111, but not 1001 or 0000.

Given a pattern P, how many triple-free binary strings does it contain?

Input

Each line of the input represents a test case, which contains the length of pattern, n ($0 < n < 31$), and the pattern P. The input terminates when $n = 0$.

The input must be read from standard input.

Output

For each test case, print the case number and the answer, as shown below.

The output must be written to standard output.

Sample Input	**Sample Output**

`4 0**1`
`5 *****`
`10 **01**01**`
`0`
	`Case 1: 2`
	`Case 2: 16`
	`Case 3: 9`

G - Grey Codes

Source file name: `grey.c`, `grey.cpp`, `grey.java`, *or* `grey.py`

We are going to generate a sequence of integers in binary. Start with the sequence:

```
0
1
```

Reflect it in the horizontal line, prepend a zero to the numbers in the top half and a one to the numbers on the bottom and you will get:

```
00
01
11
10
```

Repeat this again, and you will have 8 numbers:

```
000   0
001   1
011   3
010   2
110   6
111   7
101   5
100   4
```

The corresponding decimal values are shown on the right.

These sequences are called *reflected Gray codes* for 1, 2, and 3 bits, respectively. A Gray code for n bits is a sequence of 2^n different n-bit integers with the property that every two neighbouring integers differ in exactly one bit. A reflected Gray code is a Gray code constructed in the way shown above.

Input

The first line of input gives the number of cases, N. Then N test cases follow, each one is a line with 2 integers: n ($1 \leq n \leq 30$) and k ($0 \leq k < 2^n$).

The input must be read from standard input.

Output

For each test case, output the integer that appears in position k of the n-bit reflected Gray code.

The output must be written to standard output.

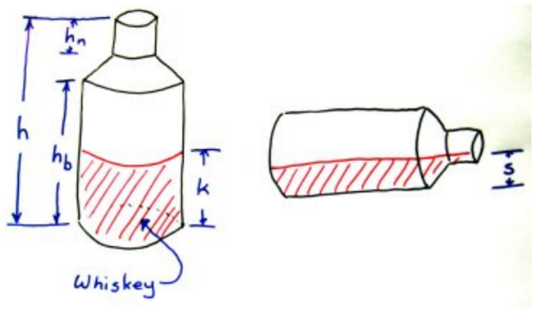
Sample Input	Sample Output
14	0
1 0	1
1 1	0
2 0	1
2 1	3
2 2	2
2 3	0
3 0	1
3 1	3
3 2	2
3 3	6
3 4	7
3 5	5
3 6	4
3 7	

H - Moonshine

Source file name: `moonshine.c`, `moonshine.cpp`, `moonshine.java`, *or* `moonshine.py`

Granny is preparing moonshine whiskey for an upcoming family reunion. Because her family is large, she needs to make several batches of whiskey, which she collects in a large metal cream can. After making, and sampling, several batches Granny stumbles and knocks the can on its side. Fortunately, the lid is tightly fixed so no whiskey is lost.

Our question to you is this: assuming that the depth of whiskey in the can was k cm with the can sitting upright, what is its depth with the can lying horizontal? The can itself is made of two cylinders: the body with height h_b and diameter d_b, and the neck with height h_n and diameter d_n. The neck and body are joined by a tapered conic shoulder such that the overall height of the can is h. The bottom and lid of the can are disks of diameter d_b and d_n, respectively.



Input

The input consists of several test cases. Each test case consists of a line containing real numbers k, h_b, d_b, h_n, d_n, h. You may assume that $100 \geq h \geq h_b + h_n$ and that $100 \geq d_b \geq d_n$.

The input must be read from standard input.

Output

For each test case, output a line containing s the depth with the can lying horizontally, rounded to two decimal places.

The output must be written to standard output.

Sample Input	**Sample Output**

`5.625 10.0 10.0 5.0 5.0 15.0`
`0 0 0 0 0 0`
	`5.00`

I - Power Signs

Source file name: `signs.c`, `signs.cpp`, `signs.java`, *or* `signs.py`

You are probably familiar with the binary representation of integers, i.e., writing a non-negative integer n as $\Sigma a_i 2^i$, where each a_i is either 0 or 1. In this problem, we consider a so called *signed binary* representation, in which we still write n as $\Sigma a_i 2^i$, but allow a_i to take on the values -1, 0, and 1. We write a signed binary representation of a number as a vector $(a_k, a_{k-1}, \ldots, a_1, a_0)$. For instance, $n = 13$ can be represented as $(1, 0, 0, -1, -1) = 2^4 - 2^1 - 2^0$.

The binary representation of a number is unique, but obviously, the signed binary representation is not. In certain applications (e.g. cryptography), one seeks to write a number n in signed binary representation with as few non-zero digits as possible. For example, we consider the representation $(1, 0, 0, -1)$ to be a better representation of $n = 7$ than $(1, 1, 1)$. Your task is to write a program which will find such a minimal representation.

Input

The input consists of several test cases, one per line. Each test case consists of a positive integer $n \le 2^{50000}$ written in binary without leading zeros. The input is terminated by a case where $n = 0$, which should not be processed.

The input must be read from standard input.

Output

For each line of input, output one line containing the signed binary representation of n that has the minimum number of non-zero digits, using the characters '-' for -1, '0' for 0 and '+' for +1. The number should be written without leading zeros. If there are several possible answers, output the one that is lexicographically smallest (by the ASCII ordering).

The output must be written to standard output.

Sample Input	Sample Output
10000 1111 10111 0	+0000 +000- ++00-

J - Macarons

Source file name: `macarons.c`, `macarons.cpp`, `macarons.java`, *or* `macarons.py`

Pierre is famous for his macarons. He makes round macarons, stored in square boxes of size 1×1, and oval-shaped macarons, stored in rectangular boxes of size 1×2 (or, rotated, in rectangular boxes of size 2×1). For the purpose of a buffet, Pierre wishes to tile a rectangular table of size $N \times M$ with the two kinds of macarons, meaning that the table must be completely full, with no empty space left. The width N of the table is small, for the guest to be able to grab the macarons easily, and the length M of the table is large, to accommodate a huge number of guests. To keep the table pretty, the orientation of macarons should always be aligned with the sides of the table.

Pierre wishes to know how many ways there are to tile the table. Can you help him?

Input

The input file contains several test case. Each test case consists of two lines, the value of $1 \leq N \leq 8$, an integer, on the first line, and the value of $1 \leq M \leq 10^{18}$, an integer, on the second line.

The input must be read from standard input.

Output

For each test case, output the total number of tilings, given modulo 10^9, on a single line.

The output must be written to standard output.

Sample Input	Sample Output

`2`
`2`
`2`
`4`
	`7`
	`71`

K - Scarecrow

Source file name: `scarecrow.c`, `scarecrow.cpp`, `scarecrow.java`, *or* `scarecrow.py`

Taso owns a very long field. He plans to grow different types of crops in the upcoming growing season. The area, however, is full of crows and Taso fears that they might feed on most of the crops. For this reason, he has decided to place some scarecrows at different locations of the field.

The field can be modeled as a $1 \times N$ grid. Some parts of the field are infertile and that means you cannot grow any crops on them. A scarecrow, when placed on a spot, covers the cell to its immediate left and right along with the cell it is on.

Given the description of the field, what is the minimum number of scarecrows that needs to be placed so that all the useful section of the field is covered? Useful section refers to cells where crops can be grown.

Input

Input starts with an integer T ($T \geq 0$), denoting the number of test cases.

Each case starts with a line containing an integer N ($0 < N \leq 1000$). The next line contains N characters that describe the field. A dot (.) indicates a crop-growing spot and a hash (#) indicates an infertile region.

The input must be read from standard input.

Output

For each case, output the case number first followed by the number of scarecrows that need to be placed.

The output must be written to standard output.

Sample Input	Sample Output

`3`
`3`
`.#.`
`11`
`...##....##`
`2`
`##`
	`Case 1: 1`
	`Case 2: 3`
	`Case 3: 0`