## Problem A: Sums

You are to write a program to compute different sums based on the given positive integer $N$.

### Input (Standard Input)

The first line of the input includes the number of test cases, $1 \leq t \leq 10000$. Each test case appears in one line containing $1 \leq N \leq 10000$.

### Output (Standard Output)

For each test case, print three space separated integers $S_1, S_2, S_3$ in one line where

- $S_1$ : the sum of first $N$ positive integer,
- $S_2$ : the sum of first $N$ positive odd integer,
- $S_3$ : the sum of first $N$ positive even integer.

### Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 3<br>1<br>11<br>111<br>1111 | 1 1 2<br>66 121 132<br>6216 12321 12432<br>617716 1234321 1235432 |

## Problem B: Musical Notes

We want to create a musical score consisting of a succession of notes. There are only two choices for the pitch of each note: each note has either a low sound or a high sound. There are also only two choices for the duration of each note: each note is either short or long. Each short note takes 1 second, and each long note takes 2 seconds. We have the following constraints:

a) The total duration of the musical score should be a given integer $n$.
b) The number of low short notes should be the same as the number of high short notes.
c) The number of low long notes should be the same as the number of high long notes.
d) There should be at least as many long notes as there are short notes.
e) Low and high notes must alternate.
f) The first note should be low.

Given an even integer $n$, we want to know the number of possible music scores satisfying the above.
For example, with $n = 6$, there are 4 possibilities:

2**2**1**1**, 2**1**1**2**, 1**2**2**1**, 1**1**2**2**

(Each note is represented by its duration, and the high notes are shown in bold.)
With $n = 8$, there is only 1 possibility: 2**2**2**2**

### Input (Standard Input)

The first line of the input includes the number of test cases $t$, $1 \leq t \leq 10000$. On each next line, a test case is specified by giving the even integer $n$, $2 \leq n \leq 100$.

### Output (Standard Output)

For each test case, print one line containing the answer to the question.

### Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 5 | 4 |
| 6 | 1 |
| 8 | 9 |
| 10 | 37 |
| 12 | 30823385424 |
| 62 | |

## Problem C: Fibonacci Sequence

Your job is to take an integer x as the input and compute $f(x) \bmod 10^9$, where $f(x)$ is the $x$-th value in the well-known Fibonacci sequence.

Note that Fibonacci sequence is defined as follows:

$$f(1) = f(2) = 1$$
$$f(k) = f(k-1) + f(k-2) \text{ for any } k > 2$$

### Input (Standard Input)

The first line of the input includes the number of test cases, $1 \le t \le 1000$. Each test case is a line containing an integer $1 \le x \le 2^{48}$.

### Output (Standard Output)

For each test case, print one line containing $f(x) \bmod 10^9$.

### Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 11<br>1<br>2<br>8<br>20<br>46<br>60<br>3749999998<br>3749999999<br>3750000000<br>3750000001<br>281474976710656 | 1<br>1<br>21<br>6765<br>836311903<br>8755920<br>499999999<br>500000001<br>0<br>500000001<br>309764667 |

## Problem D: Distinct rational numbers

You are to write a program to compute the number of distinct rational numbers $a/b$ for the given positive integer $N$ where $0 \le a \le b \le N$.

### Input (Standard Input)

The first line of the input includes the number of test cases, $1 \le t \le 10000$. Each test case comes in one line containing $2 \le N \le 10000$.

### Output (Standard Output)

For each test case, print the number of distinct rational numbers in one line.

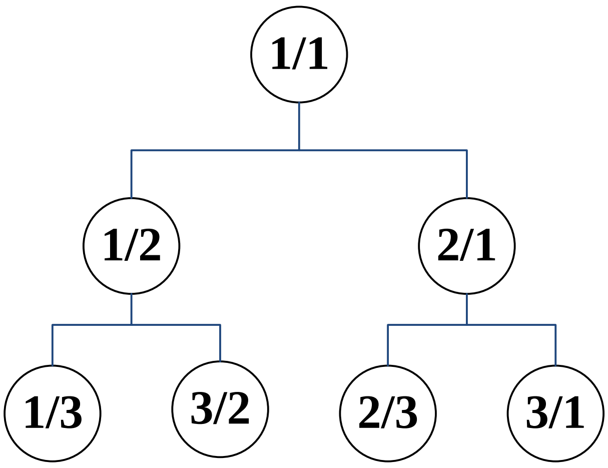
### Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 4<br>6<br>15<br>57<br>9999 | 13<br>73<br>1001<br>30393487 |

## Problem E: Sequence

We inductively label the nodes of a rooted binary tree with an infinite number of nodes as follows:

- The root is labeled by $1/1$,
- If the label of a node is $p/q$, then
    - The label of its left child is $p/(p+q)$, and
    - The label of its right child is $(p+q)/q$.



By having this tree in our hand, we define a rational sequence $a_1, a_2, a_3, \ldots$ by a breadth first traversal of the tree in such a way that nodes in the same level are visited from left to right. Therefore, we have $a_1 = 1/1$, $a_2 = 1/2$, $a_3 = 2/1$, $a_4 = 1/3$, $a_5 = 3/2$, ...

You are to write a program that gets values p and $q$ and computes an integer n for which $a_n = \mathrm{p/q}$.

### Input (Standard Input)

The first line of the input includes the number of test cases, $1 \le t \le 1000$. Each test case consists of one line. This line contains p, followed by / and then q without any space between them.

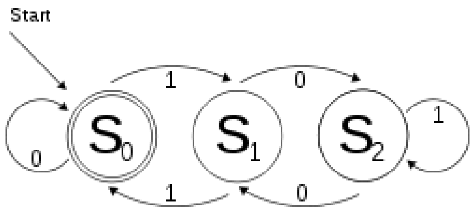
### Output (Standard Output)

For each test case, output in one line an integer n for which $a_n = \mathrm{p/q}$. It is guaranteed that in all test cases n fits in a 32-bit integer.

### Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 4<br>1/1<br>1/3<br>5/2<br>2178309/1346269 | 1<br>4<br>11<br>1431655765 |

## Problem F: History-Cleanable DFA



You have possibly heard of DFAs. In theory of computation, a deterministic finite automaton (DFA) is a finite state machine that accepts/rejects finite input strings. The figure on the right illustrates a DFA using a state diagram. In the automaton, there are three states: $\boldsymbol{S_0}$, $\boldsymbol{S_1}$, and $\boldsymbol{S_2}$ (denoted graphically by circles). The automaton takes a finite sequence of 0s and 1s as input. For each state, there is a transition arrow leading out to a next state for both 0 and 1. Upon reading a symbol, a DFA jumps from a state to another by following the transition arrow. For example, if the automaton is currently in state $\boldsymbol{S_0}$ and current input symbol is 1 then it jumps to state $\boldsymbol{S_1}$. This is formally written as $\delta(\boldsymbol{S_0}, 1) = \boldsymbol{S_1}$. Inductively, we can generalize this notation to have propositions like $\delta(\boldsymbol{S_0}, 10) = \boldsymbol{S_2}$ and $\delta(\boldsymbol{S_1}, 011010) = \boldsymbol{S_0}$. A DFA has a start state (denoted graphically by an arrow coming in from nowhere) where computations begin with, and a set of accept states (denoted graphically by a double circle) which help define when a computation is successful. So, an input string $\boldsymbol{w}$ is accepted by a DFA with starting state $\boldsymbol{q_0}$ if and only if $\delta(\boldsymbol{q_0}, \boldsymbol{w})$ is an accepting state of that DFA. The state $\boldsymbol{S_0}$ in the DFA depicted above is both the start state and an accept state. In fact, this DFA accepts only binary numbers that are multiples of 3 (including the empty string).

Given A DFA $\boldsymbol{D}$ with states $\boldsymbol{S_0}, \boldsymbol{S_1}, \dots, \boldsymbol{S_{n-1}}$, a string $\boldsymbol{w}$ is called *history-cleaner* for $\boldsymbol{D}$ if for all $i, j \in \{0, \dots, n-1\}$: $\delta(\boldsymbol{S_i}, \boldsymbol{w}) = \delta(\boldsymbol{S_j}, \boldsymbol{w})$. In other words, no matter which state the starting state is, the string $\boldsymbol{w}$ brings the DFA to a common final state. A DFA $\boldsymbol{D}$ is called *history-cleanable* if a history-cleaner string exists for $\boldsymbol{D}$. Given a DFA whose input is a sequence of 0s and 1s, your job is to find out whether it is history-cleanable or not.

## Input (Standard Input)

The first line of the input contains the single integer $t$, the number of test cases ($1 \le t \le 500$). Each test case starts with a line containing a single integer $n$, the number of states in a DFA with states $\boldsymbol{S_0}, \boldsymbol{S_1}, \dots, \boldsymbol{S_{n-1}}$ ($2 \le n \le 500$). The second line of each test case consists of $n$ space-separated integers $a_0, a_1, \dots, a_{n-1}$ and finally, the third line of each test case has $n$ space-separated integers $b_0, b_1, \dots, b_{n-1}$ ($0 \le a_i, b_i < n$) which means $\delta(\boldsymbol{S_i}, 0) = \boldsymbol{S_{a_i}}$ and $\delta(\boldsymbol{S_i}, 1) = \boldsymbol{S_{b_i}}$ (for $0 \le i < n$).

## Output (Standard Output)

For each test case, print one line containing the answer for the given DFA. If it is history-cleanable, print "YES", otherwise print "NO" (omit the quotes).

## Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 2<br>4<br>1 2 3 0<br>3 0 1 0<br>4<br>1 2 3 0<br>2 2 1 2 | NO<br>YES |

## Problem G: Partitioning a Queue

There are $n$ passengers in a single queue, waiting to enter the airplane. To avoid congestion, the queue should be partitioned into smaller parts. There are several ways to partition the queue. For instance, a queue of size 3 can be partitioned into1+2, 2+1, 1+1+1 or 3. It is easy to prove that there are $2^{n-1}$ ways to partition a queue of size $n$.

The problem becomes a little complicated, when we are not allowed to use parts whose sizes are in some given set $S$. For instance, if $S$ is the set of even numbers, a queue of size 4 can be partitioned in 3 ways, namely 1+1+1+1, 1+3 and 3+1. You're task is to count the number of ways to partition a queue of size $n$, with an additional condition that the size of no part should be in a given arithmetic sequence $\{m + ik | i = 0, 1, \ldots\}$ for the given $m, k$.

### Input (Standard Input)

The first line of the input includes the number of test cases $1 \leq t \leq 10000$. Each test case consists of three space separated integers $n, m, k$ $(1 \leq n \leq 30, 0 \leq m < k < 30)$.

### Output (Standard Output)

For each test case, print one line containing the answer to the question

### Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 3<br>10 0 2<br>15 1 4<br>28 3 7 | 55<br>235<br>18848806 |

## Problem H: Brocard Point

Label the vertices of a triangle in anticlockwise order by A, B and C. We know there is exactly one point P inside the triangle ABC such that ∠PAB = ∠PBC = ∠PCA. This point is known as Brocard point.

You are to write a program that gets the coordinates of $A$, $B$ and $C$ and computes the coordinates of the Brocard point of the triangle $ABC$.

### Input (Standard Input)

The first line of the input includes the number of test cases, $1 \leq t \leq 10000$. Each test case comes in one line containing six space separated real numbers $x_A, y_A, x_B, y_B, x_C, y_C$.

### Output (Standard Output)

For each test case, output in one line $x$-coordinate and $y$-coordinate of Brocard point rounded to five decimal places.

### Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 3<br>0 -1.3 3.4 0.5 1.1 2.3<br>0 0 3 0 0 4<br>3.1 0.2 4.3 0.4 0 0.8 | 1.40456 0.82890<br>1.56047 0.74902<br>3.87699 0.40167 |

## Problem I: RATS

A RATS sequence in base 10 is started by a positive decimal integer. From a number *i* in the sequence, the next term is generated by adding *i* to its reverse and sorting the digits. For instance, the next term after 12334444 is 55667777. Also the next term after 44556 is 111 (after dropping the leading zeros):

2334444 + 44443321 = 56777765 -> 55667777

44556 + 65544 = 110100 -> 111

According to a conjecture, each RATS sequence enters either a *repeat*, or a *chain*.
The sequence starting with 123 for example enters a repeat when 444 appears for the second time:

123, 444, 888, 1677, 3489, 12333, 44556, 111, 222, 444, 888, ...

A sequence enters a chain when a term of the form 1233*4444 or 5566*7777 appears (3* means one or more consecutive digits 3). In this case, the number of 3's and 6's increase steadily and the terms go to infinity. Below is an example:

12334444, 55667777, 123334444, 556667777, 1233334444, 5566667777, ...

Your program should identify whether a RATS sequence enters a repeat or a chain during the first *M* terms.

### Input (Standard Input)

The first line of the input includes the number of test cases, $1 \leq t \leq 10000$. Each test case consists of a line containing two integers: the number of RATS sequence terms to compute $(1 \leq M \leq 60)$ and the first number in the RATS sequence which is decimal integer with digits in increasing order and having at most 40 digits. The following terms of the RATS sequence might have more than 40 digits.

### Output (Standard Output)

For each test case, print one line as follows. If the RATS sequence enters a repeat within the first *M* term, the line follows by an uppercase letter "R" and the index of the term from which the sequence enters a repeat.

For the case of chain, the test case number is followed by the uppercase letter "C" and the index of the term from which the sequence enters a chain.

If the sequence does not enter a repeat or chain, just print the *M*th term of the sequence.

### Sample Input and Output

| Standard Input | Standard Output |
|---|---|
| 3<br>30 123<br>30 1<br>39 12222444456679999 | R 10<br>C 20<br>122233338889 |