WEEK – 1

Problem - 1

Adding Reversed Numbers

Problem code: ADDREV

The Antique Comedians of Malidinesia prefer comedies to tragedies. Unfortunately, most of the ancient plays are tragedies. Therefore the dramatic advisor of ACM has decided to transfigure some tragedies into comedies. Obviously, this work is very hard because the basic sense of the play must be kept intact, although all the things change to their opposites. For example the numbers: if any number appears in the tragedy, it must be converted to its reversed form before being accepted into the comedy play.

Reversed number is a number written in arabic numerals but the order of digits is reversed. The first digit becomes last and vice versa. For example, if the main hero had 1245 strawberries in the tragedy, he has 5421 of them now. Note that all the leading zeros are omitted. That means if the number ends with a zero, the zero is lost by reversing (e.g. 1200 gives 21). Also note that the reversed number never has any trailing zeros.

ACM needs to calculate with reversed numbers. Your task is to add two reversed numbers and output their reversed sum. Of course, the result is not unique because any particular number is a reversed form of several numbers (e.g. 21 could be 12, 120 or 1200 before reversing). Thus we must assume that no zeros were lost by reversing (e.g. assume that the original number was 12).

Input

The input consists of N cases (equal to about 10000). The first line of the input contains only positive integer N. Then follow the cases. Each case consists of exactly one line with two positive integers separated by space. These are the reversed numbers you are to add.

Output

For each case, print exactly one line containing only one integer - the reversed sum of two reversed numbers. Omit any leading zeros in the output.

Example

```
Sample input:

3
24 1
4358 754
305 794

Sample output:

34
1998
1
```

Problem - 2

Factorial

Problem code: FCTRL

The most important part of a GSM network is so called *Base Transceiver Station* (*BTS*). These transceivers form the areas called *cells* (this term gave the name to the cellular phone) and every phone connects to the BTS with the strongest signal (in a little simplified view). Of course, BTSes need some attention and technicians need to check their function periodically.

ACM technicians faced a very interesting problem recently. Given a set of BTSes to visit, they needed to find the shortest path to visit all of the given points and return back to the central company building. Programmers have spent several months studying this problem but with no results. They were unable to find the solution fast enough. After a long time, one of the programmers found this problem in a conference article. Unfortunately, he found that the problem is so called "Travelling Salesman Problem" and it is very hard to solve. If we have N BTSes to be visited, we can visit them in any order, giving us N! possibilities to examine. The function expressing that number is called factorial and can be computed as a product 1.2.3.4....N. The number is very high even for a relatively small N.

The programmers understood they had no chance to solve the problem. But because they have already received the research grant from the government, they needed to continue with their studies and produce at least *some* results. So they started to study behaviour of the factorial function.

For example, they defined the function Z. For any positive integer N, $Z(N)$ is the number of zeros at the end of the decimal form of number N!. They noticed that this function never decreases. If we have two numbers$N_1<N_2$, then $Z(N_1)$ <= $Z(N_2)$. It is because we can never "lose" any trailing zero by multiplying by any positive number. We can only get new and new zeros. The function Z is very interesting, so we need a computer program that can determine its value efficiently.

Input

There is a single positive integer T on the first line of input (equal to about 100000). It stands for the number of numbers to follow. Then there are T lines, each containing exactly one positive integer number N, 1 <= N<= 1000000000.

Output

For every number N, output a single line containing the single non-negative integer $Z(N)$.

Example

Sample Input:

```
6
```

```
3
60
100
1024
23456
8735373
```

Sample Output:

```
0
14
24
253
5861
2183837
```

Problem - 3

Small Factorials

Problem code: FCTRL2

You are asked to calculate factorials of some small positive integers.

Input

An integer t, 1<=t<=100, denoting the number of test cases, followed by t lines, each containing a single integer n, 1<=n<=100.

Output

For each integer n given at input, display a line with the value of n!

Example

Sample input:

```
4
1
2
5
3
```

Sample output:

```
1
2
120
6
```

Problem - 4

Prime Generator

Problem code: PRIME1

Peter wants to generate some prime numbers for his cryptosystem. Help him! Your task is to generate all prime numbers between two given numbers!

Input

The input begins with the number t of test cases in a single line (t<=10). In each of the next t lines there are two numbers m and n (1 <= m <= n <= 1000000000, n-m<=100000) separated by a space.

Output

For every test case print all prime numbers p such that m <= p <= n, one number per line, test cases separated by an empty line.

Example

```
Input:
2
1 10
3 5

Output:
2
3
5
7

3
5
```

Problem - 5

Transform the Expression

Problem code: ONP

Transform the algebraic expression with brackets into RPN form (Reverse Polish Notation). Two-argument operators: +, -, *, /, ^ (priority from the lowest to the highest), brackets ( ). Operands: only letters: a,b,...,z. Assume that there is only one RPN form (no expressions like a*b*c).

Input

```
t [the number of expressions <= 100]
expression [length <= 400]
[other expressions]
```

Text grouped in [ ] does not appear in the input file.

Output

The *expressions* in RPN form, one per line.

Example

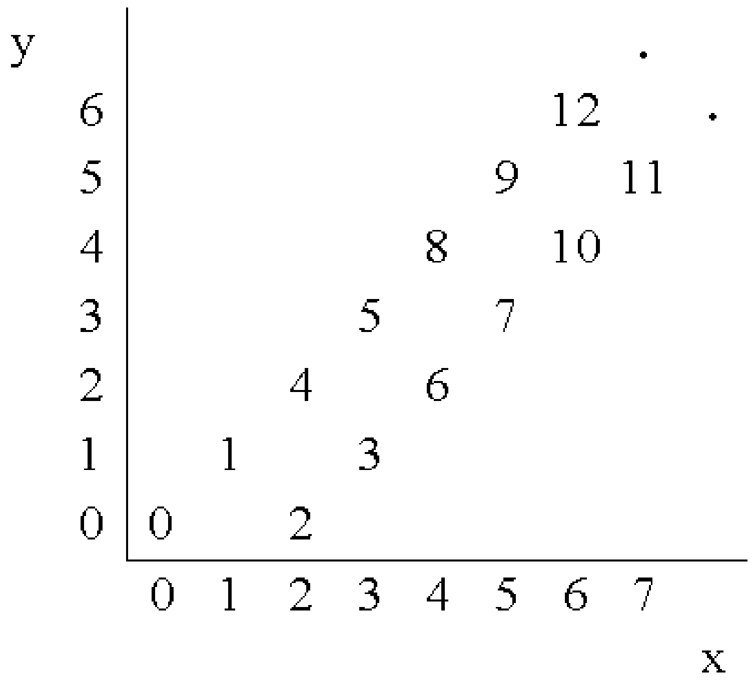
```
Input:
3
(a+(b*c))
((a+b)*(z+x))
((a+t)*((b+(a+c))^(c+d)))

Output:
abc*+
ab+zx+*
at+bac++cd+^*
```

Problem - 6

Number Steps

Problem code: NSTEPS

Starting from point (0,0) on a plane, we have written all non-negative integers 0, 1, 2,... as shown in the figure. For example, 1, 2, and 3 has been written at points (1,1), (2,0), and (3, 1) respectively and this pattern has continued.



You are to write a program that reads the coordinates of a point (x, y), and writes the number (if any) that has been written at that point. (x, y) coordinates in the input are in the range 0...10000.

Input

The first line of the input is N, the number of test cases for this problem. In each of the N following lines, there is x, and y representing the coordinates (x, y) of a point.

Output

For each point in the input, write the number written at that point or write No Number if there is none.

Example

Input:
```
3
4 2
6 6
3 4
```

Output:
```
6
12
No Number
```

Problem - 7

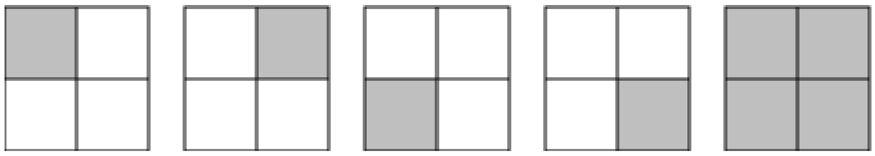
Feynman

Problem code: SAMER08F

Richard Phillips Feynman was a well-known American physicist and a recipient of the Nobel Prize in Physics. He worked in theoretical physics and also pioneered the field of quantum computing. He visited South America for ten months, giving lectures and enjoying life in the tropics. He is also known for his books "Surely You're Joking, Mr. Feynman!" and "What Do You Care What Other People Think?", which include some of his adventures below the equator.

His life-long addiction was solving and making puzzles, locks, and cyphers. Recently, an old farmer in South America, who was a host to the young physicist in 1949, found some papers and notes that is believed to have belonged to Feynman. Among notes about mesons and electromagnetism, there was a napkin where he wrote a simple puzzle: "how many different squares are there in a grid of $N \times N$ squares?".

In the same napkin there was a drawing which is reproduced below, showing that, for N=2, the answer is 5.



Input

The input contains several test cases. Each test case is composed of a single line, containing only one integer N, representing the number of squares in each side of the grid ($1 \leq N \leq 100$).

The end of input is indicated by a line containing only one zero.

Output

For each test case in the input, your program must print a single line, containing the number of different squares for the corresponding input.

Example

Input:
```
2
1
8
0
```

Output:
```
5
1
204
```

Problem - 8

To and Fro

Problem code: TOANDFRO

Mo and Larry have devised a way of encrypting messages. They first decide secretly on the number of columns and write the message (letters only) down the columns, padding with extra random letters so as to make a rectangular array of letters. For example, if the message is “There’s no place like home on a snowy night” and there are five columns, Mo would write down

```
t o i o y
h p k n n
e l e a i
r a h s g
e c o n h
s e m o t
n l e w x
```

Note that Mo includes only letters and writes them all in lower case. In this example, Mo used the character ‘x’ to pad the message out to make a rectangle, although he could have used any letter. Mo then sends the message to Larry by writing the letters in each row, alternating left-to-right and right-to-left. So, the above would be encrypted as

```
toioynnkpheleaigshareconhtomesnlewx
```

Your job is to recover for Larry the original message (along with any extra padding letters) from the encrypted one.

Input

There will be multiple input sets. Input for each set will consist of two lines. The first line will contain an integer in the range 2...20 indicating the number of columns used. The next line is a string of up to 200 lower case letters. The last input set is followed by a line containing a single 0, indicating end of input.

Output

Each input set should generate one line of output, giving the original plaintext message, with no spaces.

Example

Input:

```
5
toioynnkpheleaigshareconhtomesnlewx
3
ttyohhieneesiaabss
0
```

Output:

```
theresnoplacelikehomeonasnowynightx
thisistheeasyoneab
```

Problem - 9

Fashion Shows

Problem code: FASHION

A fashion show rates participants according to their level of hotness. Two different fashion shows were organized, one for men and the other for women. A date for the third is yet to be decided ;) .

Now the results of both fashion shows are out. The participants of both the fashion shows have decided to date each other, but as usual they have difficuly in choosing their partners. The Maximum Match dating serive (MMDS) comes to their rescue and matches them in such a way that that maximizes the hotness bonds for all couples.

If a man has been rated at hotness level `x` and a women at hotness level `y`, the value of their hotness bond is`x*y`.

Both fashion shows contain **N** participants each. MMDS has done its job and your job is to find the sum of hotness bonds for all the couples that MMDS has proposed.

Input

The first line of the input contains an integer **t**, the number of test cases. **t** test cases follow.

Each test case consists of 3 lines:

- The first line contains a single integer **N** (1 <= **N** <= 1000).
- The second line contains **N** integers separated by single spaces denoting the hotness levels of the men.
- The third line contains **N** integers separated by single spaces denoting the hotness levels of the women.

All hotness ratings are on a scale of 0 to 10.

Output

For each test case output a single line containing a single integer denoting the sum of the hotness bonds for all pairs that MMDS has proposed.

Example

Input:
```
2
2
1 1
3 2
3
2 3 2
1 3 2
```

Output:
```
5
15
```

Problem - 10

What's Next

Problem code: ACPC10A

According to Wikipedia, an arithmetic progression (AP) is a sequence of numbers such that the difference of any two successive members of the sequence is a constant. For instance, the sequence 3, 5, 7, 9, 11, 13, . . . is an arithmetic progression with common difference 2. For this problem, we will limit ourselves to arithmetic progression whose common difference is a non-zero integer.
On the other hand, a geometric progression (GP) is a sequence of numbers where each term after the first is found by multiplying the previous one by a fixed non-zero number called the common ratio. For example, the sequence 2, 6, 18, 54, . . . is a geometric progression with common ratio 3. For this problem, we will limit ourselves to geometric progression whose common ratio is a non-zero integer.
Given three successive members of a sequence, you need to determine the type of the progression and the next successive member.

Input

Your program will be tested on one or more test cases. Each case is specified on a single line with three integers (−10, 000 < a1 , a2 , a3 < 10, 000) where a1 , a2 , and a3 are distinct.
The last case is followed by a line with three zeros.

Output

For each test case, you program must print a single line of the form:
XX v
where XX is either AP or GP depending if the given progression is an Arithmetic or Geometric Progression. v is the next member of the given sequence. All input cases are guaranteed to be either an arithmetic or geometric progressions.

Example

Input:
```
4 7 10
2 6 18
0 0 0
```

Output:
```
AP 13
GP 54
```