Problem A: Two's Complement

Input File:	`a.in`
Source File:	`a.c | a.cpp | a.java`
Time Limit:	`3 seconds`

Problem Description

Computers represent numbers as a sequence of binary digits or bits. For instance the 8-bit number 00101101_2 is used to represent 45_{10} since

$$0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 45_{10}.$$

To represent negative numbers, computers actually make use of the additive inverses and assigns the most significant bit (the MSB or the leftmost bit) for the sign. This technique, called the two's complement notation, is used by the C, C++ and Java's primitive `int` datatype.

The two's complement of a value is found by first taking the one's complement (a binary number consisting of bits which is the inverse of the bits representing the original number), then incrementing that result by 1. For example, to represent the -45_{10}, which is the additive inverse of 45_{10}, as an 8-bit binary number, we first need to get the unsigned binary representation of 45_{10} then invert each bit and add 1 to the result. This yields 11010011_2 (see below).

Original Value	(45_{10})	0	0	1	0	1	1	0	1
One's Complement		1	1	0	1	0	0	1	0
Two's Complement	(-45_{10})	1	1	0	1	0	0	1	1

In the two's complement notation, a binary value with a 0 in the MSB position is considered positive and a binary value with a 1 in the MSB position is considered negative. This introduces one important constraint of this notation: Since the MSB is used to indicate the sign, half of the bit patterns lose their positive association. In fact, for an n-bit two's complement number z, it is easy to see that $-2^{n-1} \leq z \leq 2^{n-1} - 1$. This is the reason why the `int` datatype, which employs a 32-bit two's complement notation, can only represent numbers from -2,147,483,648 to 2,147,483,647.

Your task is to write a program that reads several integers and converts them to two's complement binary notation with specified number of bits.

Input Format

The input starts with a positive integer $k \leq 50$, which denotes the number of cases. This is then followed by k lines where each line contains integer z, the number you need to convert into two's complement, and n, the number of bits that will be used for representing the number in binary. These two integers are separated by a single space. Assume that $-2^{31} \leq z \leq 2^{31} - 1$ and $0 < n \leq 32$.

Output Format

The output for each case begin with a line containing the word `CASE` followed by a single space, the case number, and a colon (:). Following the case label, print equivalent value of the x in two's complement using n bits. If z cannot fit in an n-bit two's complement number, print the word `IMPOSSIBLE`.

Sample Input

```
3
45 8
-45 8
45 6
```

Sample Output

```
CASE 1:
00101101
CASE 2:
11010011
CASE 3:
IMPOSSIBLE
```

Solution

Solution Author: Glenn Fabia
Institution: Ateneo de Naga University
Contact: gfabia@gbox.adnu.edu.ph

This problem does not have an editorial. Please contact the problem setter via the contact information given above.

Problem B: RANDU

Input File: `b.in`
Source File: `b.c | b.cpp | b.java`
Time Limit: `3 seconds`

Problem Description

Many random-number generators in use today are *Linear Congruential Generators (LCGs)*, introduced by Lehmer (1951). A sequence of integers Z_1,Z_2, ... is defined by the recursive formula

$$Z_i = (aZ_{i-1} + c)(\texttt{mod}\ m)$$

where m (the modulus), a (the multiplier), c (the increment), and Z_0 (the seed or starting value) are nonnegative integers. Also, $0 < m$, $a < m$, $c < m$, and $Z_0 < m$. That is, to obtain Z_i, divide $aZ_{i-1} + c$ by m and let Z_i be the *remainder* of this division.

For example, the LCG $Z_i = (5Z_{i-1} + 3)(\texttt{mod}\ 16)$ with $Z_0 = 7$ generates the following sequence of numbers:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Z_i	7	6	1	8	11	10	5	12	15	14	9	0	3	2	13	4	7	6	1	8	11

Note that Z_i repeats. The length of the cycle is referred to as the *period* of a generator. For LCG with $0 \leq Z_i \leq m - 1$, the period would evidently be at most m. If it is in fact equal to m, we say that the LCG has a *full period*.

The parameters m, a, c, and Z_0 are typically chosen to optimize computational and space efficiency as well as observed certain statistical properties. For instance, a good candidate for the multiplier a is an integer of the form $2^l + j$ (so that multiplication of Z_{i-1} by a is replaced by a shift and j adds). Also, the modulo, m, is usually chosen to be very large for it is much desirable to have LCGs with long periods. Immediately, one can notice that a candidate for m would be an integer of the form 2^b, with b being the number of bits on the computers. The generator known as RANDU is of this form ($m = 2^{31} = 2,147,483,648$, $a = 2^{16} + 3 = 65,539$, and $c = 0$). Note that in RANDU, $b = 31$ since in most 32-bit computers (and compilers) the leftmost bit is used to represent the sign.

Your task is to write a program that implements the RANDU generator described above.

Input Format

The input starts with a positive integer $k \leq 50$, which denotes the number of cases. This is then followed by k lines where each line contains integer Z_0, the seed value, and n, the number of random numbers to generate. These two integers are separated by a single space. Assume that $0 \leq Z_0 \leq 2^{31}$ and $0 < n \leq 2^{16}$.

Output Format

The output for each case begins with a line containing the word `CASE` followed by a single space, the case number, and a colon (:). Below the case label are the corresponding Z_i's, $i = \{1, \ldots, n\}$ for the case.

Sample Input

```
3
1 5
1 10
1772212344 20
```

Sample Output

```
CASE 1:
65539
393225
1769499
7077969
26542323
CASE 2:
65539
393225
1769499
7077969
26542323
95552217
334432395
1146624417
1722371299
14608041
CASE 3:
224227688
427840568
548994216
1590883832
309387752
423273912
1902637352
1163907960
597129832
1697541944
516115880
```

703719672

1724758760

1867591864

2125173288

237680248

1626874728

1179675192

1026113192

1982053368

Solution

Solution Author: Glenn Fabia
Institution: Ateneo de Naga University
Contact: gfabia@gbox.adnu.edu.ph

This problem does not have an editorial. Please contact the problem setter via the contact information given above.

Problem C: Sharkovski Successor

Input File:	`c.in`
Source File:	`c.c | c.cpp | c.java`
Time Limit:	`1 second`

Problem Description

In a 1964 paper on continuous mappings of the reals into the reals, Alexandr Sharkovski used the following ordering of the positive integers:

$3 \blacktriangleleft 5 \blacktriangleleft 7 \blacktriangleleft 9 \blacktriangleleft ... \blacktriangleleft 3 \cdot 2 \blacktriangleleft 5 \cdot 2 \blacktriangleleft 7 \cdot 2 \blacktriangleleft ... \blacktriangleleft 3 \cdot 2^2 \blacktriangleleft 5 \cdot 2^2 \blacktriangleleft ... \blacktriangleleft 2^3 \blacktriangleleft 2^2 \blacktriangleleft 2 \blacktriangleleft 1$

As Ciesielski and Pogoda (2008) describe it:

"First come the odd numbers, beginning with 3, arranged in increasing order. This sequence is repeated with each odd integer multiplied by 2. The initial sequence is again repeated with each odd integer multiplied by 2^2, and so on. The terminal sequence consists of the non-negative powers of 2 arranged in decreasing order (note that $1 = 2^0$)."

Write a program that reads a list of up to 254 positive integers with values less than 65,535 separated by white space and terminated by the number 0. For each positive integer in the list, the program should output the next number in the Sharkovski ordering (the 'Sharkovski successor'). Note that the number 1 has no Sharkovski successor, but for this problem, we will let the successor of the number 1 be the number 3.

Input Format

The input is a list of up to 254 positive integers with values less than 65,535 separated by white space and terminated by 0.

Output Format

The output is a list of the Sharkovski successors of the input positive integers in the corresponding order. The output integers are separated by exactly one space.

Sample Input

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 49152 63488 0
```

Sample Output

```
3 1 5 2 7 10 9 4 11 14 13 20 15 18 17 8 19 22 21 81920 67584
```

Solution

Solution Author: Joel Noche
Institution: Ateneo de Naga University
Contact: jnoche@gbox.adnu.edu.ph

```c
language: C

/* http://bit-player.org/2008/the-end-of-the-number-line */

#include <stdio.h>

unsigned int s(unsigned int k)
{
        if (k & 1) /* k is odd */
                return k + 2;
        else
                if (k == 2)
                        return 1;
                else
                        return 2 * s(k/2);
}

unsigned int main(void)
{
        unsigned int i[255], j = 0, n;
        FILE *fp;

        if ((fp = fopen("x.in", "rt")) == NULL)
                return 1;
        do
        {
                fscanf(fp, "%u", &n);
                i[j++] = n;
        }
        while (n != 0);
        fclose(fp);
        n = j - 1;
        /* n is the number of positive integers in the input */

        for (j = 0; j < n; j++)
                printf("%u ", s(i[j]));
        printf("\n");
        return 0;
}
```

Problem D: String Substitution

Input File:	`d.in`
Source File:	`d.c | d.cpp | d.java`
Time Limit:	`1 second`

Problem Description

Write a program that uses the following algorithm:

1. Read an input string composed of digits, that is, characters belonging to the set $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
2. Read the string from left to right and replace the first three consecutive identical digits (if any) with one digit whose value is the sum of the three digits modulo 10.
3. Repeat step 2 until there are no more substrings of three consecutive identical digits.
4. Display the resulting string as output.

Input Format

The input starts with an integer N (with $0 < N < 255$). This is followed by N lines of input cases. Each input case is a string of d digits (with $0 < d < 255$).

Output Format

The output has N lines of strings: the output strings in the same order as their corresponding input strings.

Sample Input

```
3
0
01234567890011223344556677889 9
000001113399774441114555222999222222222111111111
```

Sample Output

```
0
01234567890011223344556677889 9
0123456789
```

Solution

Solution Author: Joel Noche
Institution: Ateneo de Naga University
Contact: jnoche@gbox.adnu.edu.ph

```
language: C

#include <stdio.h>

char s[255];
typedef enum {false, true} bool;

bool ss(void)
{
	unsigned int j = 0, k;
	bool r = false;

	while (s[j])
	{
		if (s[j] == s[j+1] && s[j] == s[j+2])
		{
			s[j] = (char)(((unsigned int)(s[j]-'0') * 3 % 10) + '0');
			k = j;
			while (s[k++])
				s[k] = s[k+1];
			k = j;
			while (s[k++])
				s[k] = s[k+1];
			r = true;
		}
		j++;
	}
	return r;
}

unsigned int main(void)
{
	unsigned int i, n;
	FILE *fp;

	if ((fp = fopen("y.in", "rt")) == NULL)
		return 1;
	fscanf(fp, "%u", &n);
	for (i = 0; i < n; i++)
	{
		fscanf(fp, "%s", s);
		while (ss());
```

```
43                printf("%s\n", s);
44        }
45        fclose(fp);
46        return 0;
47 }
```

Problem E: Discover Weekly

Input File:	`e.in`
Source File:	`e.c | e.cpp | e.java`
Time Limit:	`2 seconds`

Problem Description

Early 2013, Spotify, the music app, launched a feature called, "Discover Weekly". Every week, each user would receive a recommended playlist composed of 30 songs based on the songs being consumed by the user for the previous weeks. According to Matt Ogle, then lead engineer of Spotify, the playlist should be a mix of familiar and new songs - Spotify's RD team discovered that if the playlist is too familiar, engagement with the playlist fell as users felt they were being fed the same song each week. If the playlist is too new, engagement also dropped as the users could not identify with the music being recommended.

Spotify's solution was to base their Discover Weekly feature on what they call a 'Collaborative Filter'. "Each week, Spotify bots hunt through several billion playlists from users around the world to see what songs are typically grouped together" (Hit Makers, The Science of Popularity in an Age of Distraction, Derek Thompson).

If user Alpha listens to songs A, B and C, and the bots find that songs A, B, and C are typically seen in other users' playlists that also contain song K, then song K would appear in user Alpha's Discover Weekly playlist. Songs in the discover weekly playlist are arranged in non-ascending order according to how frequent they appear together in a playlist containing the user's preferred songs. Songs with similar frequencies are arranged in alphabetical order.

Input Format

The input contains an integer T, where $0 < T < 100$, followed by T test cases. Each test case is composed of a string S, representing the name of the user, followed by three space separated characters, $C1$, $C2$, and $C3$, where $[C = A \ldots Z]$ denoting the user's top three most-listened-to songs for the week. It is followed by an Integer P, where $0 < P < 100$, denoting the number of random playlists from which the discover weekly playlist would be based on. Each case P is composed of an integer Q, representative of the number of songs in the playlist, followed by Q number of songs, C's.

Output Format

From the user's top three songs, compose a discover weekly playlist of 5 songs. If the data isn't enough to create a playlist of 5 songs, insert an asterisk ($*$) placeholder. Output format should be the name of the user and the user's discover weekly playlist separated by a colon (:). Assume it is guaranteed that a playlist of at least 1 song can be created from the input.

Sample Input

```
3

Yuubi
A C F
3
```

```
5 A C F X D
6 A C F V B X
6 A C F V H U

Armi
X G J
3
6 X G J K R M
6 X H U P O L
6 G T Y A U Q

Herschel
T A N
5
10 T A X M Y U I K L Q R J S
5 A X M P T N
7 N X M A T R J
4 T N A E
5 N U A T O
```

Sample Output

```
Yuubi : V X B D H
Armi : K M R * *
Herschel : M X E J P
```

Solution

Solution Author: Neithan Casano
Institution: Ateneo de Naga University
Contact: jonathancasano@gmail.com

The solution is straightforward as the steps have been outlined in the problem description. Since songs are represented by letters $A...Z$, a 26-index array could be prepared and incremented per song that gets read as input.

Afterwhich, the indexes could be sorted in non-ascending order and the top five entries could just be printed out one after the other.

Gotchas:

1. The sorting should preserve lexicographical order. In the case of a tie between two songs, the letter that should first be printed should be the one that comes first in lexicographical ordering.
2. If the count of the songs does not reach 5, asterisks should be printed in place of letters.

Problem F: J'anne and Approximations

Input File:	`f.in`
Source File:	`f.c | f.cpp | f.java`
Time Limit:	`1 second`

Problem Description

Yesterday, I was trying to tutor my younger sister J'anne on finding the square root of a number.

I would ask, *"Alright, J'anne. What's the square root of* 9*?"*. She would give me haphazard answers, *"One thousand!"*, she would blurt out proudly. She was doing it on purpose; trying to mess with me. This kept going for a few hours.

"Square root of 10*?"*, *"*450*!"*. *"Square root of* 14*?"*, *"*729*!!"*.

I decided to change my strategy. *"Hey, J'anne. What if I told you I can get the correct square root of a number starting from whatever insane incorrect answer you come up with?"*.

She seemed interested. Her voice sounded intrigued, *"You can do that? Even if I say the square root of* 25 *is* $1,000,000$*?"*.

"Yes", I answered, *"There is a series of steps you just have to repeat to turn* $1,000,000$ *to a* 5 *- and it would be correct all the time!"*.

"Yeah, right. Like Math is supposed to be THAT easy", she responded with a tone.

I got my notebook and calculator, followed my steps, and within a few lines, given 25, I was able to get 5, its correct square root, from $1,000,000$ her over-the-board estimate.

J'anne snagged my notebook and looked at it deeply, as if trying to make sense of what just happened. After a few moments, her gaze swiftly shifted towards me...

"This... teach me this! I need to know this.", she insisted, in a cold commanding tone.

One of the ways to get the square root of a number x is to first come up with a guess y.

1. Divide the input number x, by the initial guess y, to acquire a quotient z
2. Get the average of z and y, to acquire y', your new guess.
3. Divide x by the new guess y' to get a new z'
4. Continue repeating steps 2 and 3 until the difference between $y' * y'$ and x is less than 0.0001
5. Output y' - this is now the correct square root

Say you're looking for the square root of 9 and you start with a guess 5. The above steps would generate this list of guesses.

- *Step 1* - $9/5 = 1.80000$
- *Step 2* - $\text{average}(5, 1.80000) = 3.40000$
- *Step 3* - $9/3.40000 = 2.65000$

- *Step 2* - average$(2.65000, 3.40000) = 3.02500$
- *Step 3* - $9/3.02500 = 2.97500$
- *Step 2* - average$(2.97500, 3.02500) = 3$
- *Step 3* - $9/3 = 3.00000$
- *Step 4* - $3.00000 * 3.00000 = 9(9 - 9$ is less then $0.0001)$
- *Step 5* - output 3.00000

Input Format

The input consists of an integer T, where $0 < T < 100$, followed by T test cases. Each test case is composed of a pair of integers N and J, the number you would have to get the square root of and J'anne's weird estimate, respectively. Operate under the following constraints: $0 < N < 100$ and $0 < J < 1,000,001$.

Output Format

Your program must output all guesses (y') leading to the correct square root. The guesses should be truncated off (not rounded off) to the nearest ten thousandths.

Sample Input

```
3
9 5
25 10
7 13
```

Sample Output

```
3.40000
3.02352
3.00009
3.00000

6.25000
5.12500
5.00152
5.00000

6.76923
3.90166
2.84788
2.65292
2.64576
```

Solution

Solution Author: Neithan Casano
Institution: Ateneo de Naga University
Contact: jonathancasano@gmail.com

This process of getting square roots is popularly known as Newton's square root method. Below is a solution written in Racket.

```
language: Racket

(define (good-enough? guess x)
    (< (abs (- x (* guess guess))) 0.0001)
)

(define (average x y)
    (/ (+ x y) 2)
)

(define (improve guess x)
    (display (average (/ x guess) guess))
    (newline)
    (average (/ x guess) guess)
)

(define (nsqrt guess x)
    (if (good-enough? guess x)
        guess
        (nsqrt (improve guess x) x))
)

(display (nsqrt 1000000.0 100.0))
```

Problem G: RPN Calculator

Input File:	`g.in`
Source File:	`g.c | g.cpp | g.java`
Time Limit:	`1 second`

Problem Description

Reverse Polish notation (RPN), also known as Polish postfix notation or simply postfix notation, is a mathematical notation in which operators follow their operands. The notation is a way of expressing arithmetic expressions that avoid the use of brackets (or parentheses) to define priorities for evaluation of operators.

In ordinary notation, for example, we write $(4 + 5) \times (2 - 3)$ and the brackets tell us that we add 4 to 5, then subtract 3 from 2, and multiply the results together, giving a final value of -9. In RPN, the numbers and operators are listed one after another, and an operator always acts on the most recent numbers in the list.

In RPN, the expression above will be $45 + 23 - \times$. We apply $+$ to 4 and 5 resulting to 9, $-$ to 2 and 3, resulting to -1 and $\times$ to 9 and -1, resulting to a final value of -9.

Input Format

Each line in the input file is a mathematical expression in Reverse Polish Notation. A mathematical expression contains numbers and the following operators: $+$ (addition), $-$ (subtraction), $\times(multiplication)$,$/(division)$, (exponentiation), and $\sim$ $(unaryminusornegation)$.

Output Format

For each input line, print the result of evaluating the expression. Express the result as a real number rounded to 2 decimal places.

Sample Input

```
4 5 + 2 3 - x
12 3 4 + -
1 20 3 4 x + -
5 3 +
40 2 ~ /
15 7 1 1 + - / 3 x 2 1 1 + + -
```

Sample Output

```
-9.00
5.00
-31.75
4.74
-2.00
5.00
```

Solution

Solution Author: Rey Herman Vidallo
Institution: Ateneo de Naga University
Contact: rvidallo@gbox.adnu.edu.ph

This is a classic data structure problem on stacks. The basic algorithm is to push operands to the stacks and if an operator op is encountered then pop operand y, pop operand x and evaluate z = x op y. push z then to stack. if operator is unary ( ), pop one operand only. at the end of the input, the value at top of stack is the final result.

```
language: C++

#include <iostream>
#include <sstream>
#include <fstream>
#include <stack>
#include <cstring>
#include <cmath>
#include <iomanip>

using namespace std;

int main(){
    ifstream infile;
    int n;
    string line, inp;
    stack<float> S;
    float x, y, r;

        infile.open("prob1.in");
        while (getline(infile, line)){
            istringstream ss(line);
            while (ss >> inp)
                if ((inp != "+") && (inp != "-") &&
                (inp != "x") && (inp != "/") && (inp != "^") &&
                (inp != "~"))
                    S.push(stof(inp));
                else if (inp=="+"){
                    y = S.top(); S.pop();
                    x = S.top(); S.pop();
                    r = x + y;
                    S.push(r);
                }
                else if (inp=="-"){
                    y = S.top(); S.pop();
                    x = S.top(); S.pop();
                    r = x - y;
```

```
                S.push(r);
            }
            else if (inp=="x"){
                y = S.top(); S.pop();
                x = S.top(); S.pop();
                r = x * y;
                S.push(r);
            }
            else if (inp=="/"){
                y = S.top(); S.pop();
                x = S.top(); S.pop();
                r = x / y;
                S.push(r);

            }
            else if (inp=="^"){
                y = S.top(); S.pop();
                x = S.top(); S.pop();
                r = pow(x,y);
                S.push(r);
            }
            else if (inp=="~"){
                x = S.top(); S.pop();
                r = -1*x;
                S.push(r);
            }
            cout.setf(ios::fixed);
            cout << setprecision(2) << S.top() << endl;
        }
        infile.close();
  return 0;
}
```

Problem H: (1,2)-Ulam #s

Input File:	`h.in`
Source File:	`h.c | h.cpp | h.java`
Time Limit:	`3 second`

Problem Description

We define the (1,2)-Ulam numbers by setting $u_1 = 1$ and $u_2 = 2$. Furthermore, after determining whether the integers less than n are Ulam numbers, we set n equal to the next Ulam number if it can be written uniquely as the sum of two different Ulam numbers.

Thus, the next Ulam number, u_3, after the initial 1, 2 is obviously 3 since $3 = 1 + 2$. The next Ulam number, u_4, is 4 since $4 = 1 + 3$. 5 is not an Ulam number since 5 could be represented in two ways: $5 = 1 + 4$ and $5 = 2 + 3$. Therefore, the next Ulam number, u_5, is 6 because $6 = 2 + 4$.

The first 10 (1-2)-Ulam numbers are $1, 2, 3, 4, 6, 8, 11, 13, 16$, and 18. Because, there are infinitely many (1,2)-Ulam numbers, we want to find u_i, the ith (1,2)-Ulam number.

Input Format

The input file contains several input integers i , where $1 \leq i \leq 500$ one integer per line.

Output Format

For each input integer i, print in a new line the integer i, followed by a space, followed by the ith (1,2)-Ulam number.

Sample Input

```
5
10
1
20
233
45
```

Sample Output

```
5 6
10 18
1 1
20 69
233 2249
45 221
```

Solution

Solution Author: Rey Herman Vidallo
Institution: Ateneo de Naga University
Contact: rvidallo@gbox.adnu.edu.ph

The solution is to go through all integers x from 3 to infinity. If x is an Ulam number, add x to the list of Ulam numbers

To check if x is an Ulam number, perform an exhaustive search on all previous Ulam numbers. if any 2 previous Ulam numbers $<$ x adds up to x, increment addcount x is an Ulam number only if addcount $==$ 1 after the exhaustive search.

```
language: C++

#include <iostream>
#include <fstream>

using namespace std;

int main(){

        ifstream infile;

        int ulam[500] = {0};
        int u1 = 1, u2 = 2, i;
        int x = 3;

        int input;

        ulam[0] = u1; ulam[1] = u2;
        i = 2;
        // generate the first 500 ulam numbers
        in the sequence to speed up processing
        do{
           int add_count = 0;
           for (int j=0; j < i; j++)
               for (int k = j+1; k < i; k++){ //
                    if (ulam[j] + ulam[k] == x)
                        add_count++;
               }
            if (add_count==1){
                ulam[i] = x;
                i++;
            }
            x++;
        }while (i < 500);

```

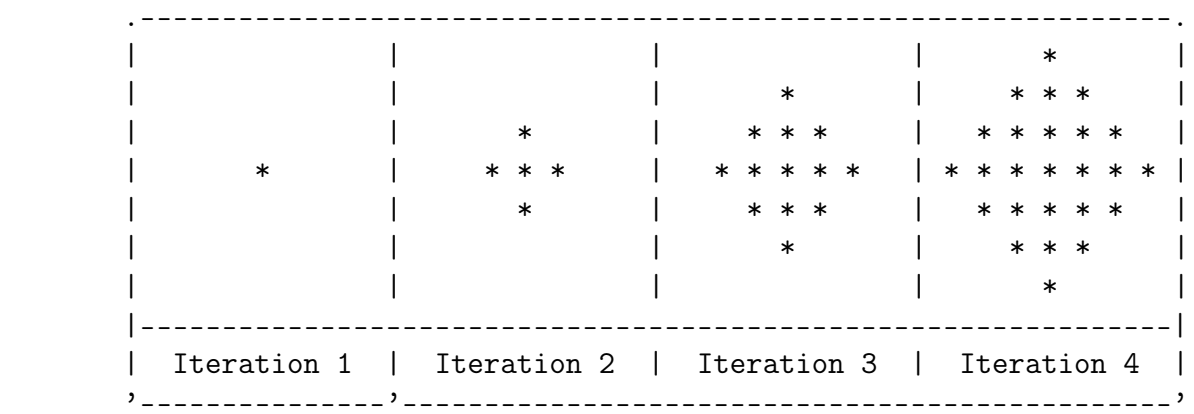
```
36          infile.open("prob2.in");
37          while (infile >> input){
38              cout << input << " " << ulam[input-1] << endl;
39          }
40          infile.close();
41
42      return 0;
43
44      // line 24: check if x is the ith Ulam number, ulam[i], by
45      exhaustively checking if for any ulam[j] and ulam[k]
46      where j < i, k < i, ulam[j] + ulam[k] == x
47  }
```

Problem I: Mitch Diamond

Input File:	`i.in`
Source File:	`i.c | i.cpp | i.java`
Time Limit:	`1 second`

Problem Description

Mitch likes to collect small rocks and arrange them in a certain manner. At first, he starts with putting one rock. In the next iterations, he would put rocks in the manner shown in the figure below.



Given a number n, your task is to find out how many rocks Mitch would need to produce the resulting pattern on the nth iteration.

Input Format

The input starts with an integer N, followed by N integers. The value of N is positive and will not exceed $1,001$. Each of the N integers are non-negative numbers and will not exceed $1,000$.

Output Format

For each of the N integers, output the number of rocks required to produce the rock pattern of Mitch.

Sample Input

```
3
0
2
4
```

Sample Output

```
0
5
25
```

Solution

Solution Author: Allan Sioson
Institution: Ateneo de Naga University
Contact: allan.sioson@gmail.com

The sequence is created by adding multiples for 4 to N, where N is the input integer.

1+4=5
5+8=13
13+12=25
25+16=41
41+20=61
61+24=85
and so on

The closed formula is as follows: (N+N) * (N-1) + 1

```
language: C++

# include <iostream>
using namespace std;

int main() {
  int N = 0;
  cin >> N;
  for (int i = 0; i < N; i++) {
    int n;
    cin >> n;
    cout << (1 + (n - 1) * (n + n)) << endl;
  }
  return 0;
}
```

Problem J: Work Together

Input File:	`j.in`
Source File:	`j.c | j.cpp | j.java`
Time Limit:	`1 second`

Problem Description

Fritz can finish painting an entire building in 5 days. Glenn can finish the same work in 5 days. When they work together, they can finish the same task in 2.5 days.

Similarly, Bax can finish painting an entire building in 4 days, Mitch can finish the same work in 4 days, and John can finish the same work in 2 days. They finish the work in 1 day when they all work together.

Write a program that determines the number of days a set of individuals can finish one particular task if they work together given the estimate of time each can finish the same task individually.

Input Format

The input starts with the number of cases N, followed by N input lines. Each input line starts with the number of individuals n followed by n estimated number of days each individual could finish the same work independently.

The value of N cannot exceed 100. The number of individuals cannot exceed 10. The number of days cannot exceed 30 days. All the numbers are positive.

Output Format

For each input line, output the number of days it would take the group to finish the job if they work together. The output should be rounded to at least 3 decimal digits.

Sample Input

```
3
2 5 5
3 4 4 2
2 3 7
```

Sample Output

```
2.500
1.000
2.100
```

Solution

Solution Author: Allan Sioson
Institution: Ateneo de Naga University
Contact: allan.sioson@gmail.com

Represent the amount of work done in terms of ratios i.e. 1/18 that is, 1 whole work can be finished in 18 units of time (days, weeks, years etc.)

Add the ratios. The reciprocal of the sum will give the units of time when the work can be finished together.

```
language: C++

# include <cstdio>
# include <iostream>
using namespace std;

int main() {
  int N;
  cin >> N;
  for (int i = 0; i < N; i++) {
    int n;
    cin >> n;
    long double sum = 0;
    for (int j = 0; j < n; j++) {
      long double x;
      cin >> x;
      sum += 1 / x;
    }
    printf("%2.3Lf\n", (1 / sum));
  }
  return 0;
}
```