Homework 2

1. Write a program that prompts for and reads a floating-point value. The program prints the whole part on one line and the decimal (fraction) part on a second line.
   For example, if the input value is 123.456, it would print the output:
   the input value is 123.456
   the whole part is 123
   the decimal(fraction) part is 0.456

2. Write a program that asks the user to enter two numbers, obtains the two numbers from the user and prints the sum, product, difference, and quotient of the two numbers.

	Number 1	Number 2
Case I	int	int
Case II	float	float
Case III	double	double
Case VI	int	float

   Consider and discuss the results.

3. Write a program that accepts an integer between 7 and 9 digits long.
   a. Extracts and prints the third-rightmost digit of the input data.
   b. Writes the integer with commas between every third digit starting from the right.

   Example:
   Input: 12345678
   Output:
   The third-rightmost digit of the input data is 6
   The input data with commas between every third digit is 12,345,678

4. Write a program to calculate the diameter, the circumference, and the area of a circle with a radius of 6.75.

   Assign the radius of float variable, and then output the radius with an appropriate message. Declare a named const PI with the value 3.14159.

   The program should output the diameter, the circumference, and the area, each on a separate line. **Print each value to five decimal places within a total field width(欄位) of 10**.

Note1: **when the compiler reads this specifier: %10.5f**

i. **The compiler prepares 10 columns to output this real number with the five right-most columns for the fraction part.**
ii. **If the real number has less than five digits in the fraction part, the compilers pads the remaining columns with zero.**
iii. **The 6th column from the right is the decimal point.**
iv. **The remaining four columns are the integer part. If the real number has less than four digits in the integer part, the output is padded with blank on the left.**

%10.5f 的意思: 這個 format 對應的 data 其輸出格式如下:
留十個欄位來輸出實數，其中後五個欄位為小數點後面的位數，不足五位則在後方補「0」，後面數來第六位數是小數點，剩下四位放整數，整數不足四位則在前方插入空白。

Note2: Be sure to include appropriate comments in your program, choose meaningful identifier.

5. The effective resistance of a parallel circuit with five parallel resistances is given by:

$$R = \frac{1}{1/R1 + 1/R2 + 1/R3 + 1/R4 + 1/R5}$$

Read these five resistances from the keyboard and calculate the effective resistance *R*.

6. Solve a set of simultaneous equations:

$$ax + by = c$$
$$dx + ey = f$$

Input data: six real numbers.
Formulas for the solution:

$$x = \frac{ce - bf}{ae - bd}$$

$$y = \frac{af - cd}{ae - bd}$$

Output all the input values *a, b, c, d, e,* and *f* and the computed values for

x and y.

7. 拈（Nim）這種遊戲遊戲**(沒有程式設計基礎的同學，以說明方式來解決．有程式設計基礎的同學請儘量以程式解決．同學們可以互相玩玩看!!)**

規則

1. n堆石子，石子的個數分別為k_1，k_2，…，k_n；
2. 遊戲者為 A 和 B 兩位，A 和 B 輪流取石子，但由 A 先取；
3. 取石子的時候只能從任何一堆的石子中取 1 個或多個石子；
4. 拿到最後一個石子的遊戲者為勝，並且遊戲過程中雙方都採取最好的策略。

有一個名詞稱為”**輸的狀態**(losing position)”，其意思是說遊戲者從此狀態玩下去，一定會輸，假如雙方都採取最好的策略。另外一個名詞稱為”**贏的移動**(winning move)”，其意思是說這一步走下去，將留給對手**輸的狀態**。你的任務就是設計一個程式決定任一給予的狀態有多少**贏的移動**。

接著提供一個定理，此定理可作為程式設計的參考。

假設有一個狀態，其n堆石子的個數分別為k_1，k_2，…，k_n，並以**二進制來表示**，則此遊戲處於**輸的狀態**，若且唯若，每一位元都包含偶數個“1”，亦即所有k_i，$1 \le i \le n$，的 xor 為 0。

例如某一狀態有 3 堆石子，$k_1 = 7$，$k_2 = 11$，$k_3 = 13$。將k_1，k_2，和 k_3表示成二進制，如下所示

0111
1011
1101

很明顯，上面的數值，左邊 3 個位元，其 “1” 的個數均為偶數，但是最右邊的位元，其 “1” 的個數為奇數，因此此遊戲不是處於**輸的狀態**。

若從第 3 堆石子取走 1 個石子，k_3從 13 變為 12，那麼每一位元都包含偶數個 “1”，此遊戲便處於**輸的狀態**。所謂**贏的移動**，便是將**輸的狀態**留給對手。上面的例子除了從第 3 堆石子取走 1 個石子之外，也可以從第 1 堆或第 2 堆取走 1 個石子，均可使此遊戲處於**輸的狀態**。因此此狀態有 3 個**贏的移動**。

為了測試程式，輸入的資料將包含多組狀態，每一組狀態的第一列為石子的堆數n，$1 \le n \le 1000$。下一列將包含n個正整數，$1 \le k_i \le 1{,}000{,}000{,}000$，$1 \le i \le n$，代表$n$堆石子的個數，並以空格格開。最後一列為 0，亦即$n = 0$，表示輸入結束。
對應每一組輸入狀態，請輸出其**贏的移動**的個數。

輸入範例
3
7 11 13
2
1000000000 1000000000
0

輸出範例
3
0

7. Nim:

In this problem, you may write out the program or just describe the method to solve the game. You may play this game each other.

Nim is a 2-player game featuring several piles of stones. Players alternate turns, and on his/her turn, a player's move consists of removing *one or more stones* from any single pile. Play ends when all the stones have been removed, at which point the last player to have moved is declared the winner. Given a position in Nim, your task is to determine how many winning moves there are in that position.

A position in Nim is called "losing" if the first player to move from that position would lose if both sides played perfectly. A "winning move," then, is a move that leaves the game in a losing position. There is a famous theorem that classifies all losing positions. Suppose a Nim position contains n piles having $k_1, k_2, \ldots, k_n$ stones respectively; in such a position, there are $k_1 + k_2 + \ldots + k_n$ possible moves. We write each k_i in binary (base 2). Then, the Nim position is losing if and only if, among all the k_i's, there are an even number of 1's in each digit position. In other words, the Nim position is losing if and only if the *xor* of the k_i's is 0.

Consider the position with three piles given by $k_1 = 7$, $k_2 = 11$, and $k_3 = 13$. In binary, these values are as follows:

111
1011
1101

There are an odd number of 1's among the rightmost digits, so this position is not losing. However, suppose k_3 were changed to be 12. Then, there would be exactly two 1's in each digit position, and thus, the Nim position would become losing. Since a winning move is any move that leaves the game in a losing position, it follows that removing one stone from the third pile is a winning move when $k_1 = 7$, $k_2 = 11$, and $k_3 = 13$. In fact, there are exactly three winning moves from this position: namely removing one stone from any of the three piles.

Input

The input test file will contain multiple test cases, each of which begins with a line indicating the number of piles, $1 \leq n \leq 1000$. On the next line, there are n positive integers, $1 \leq k_i \leq 1,000,000,000$, indicating the number of stones in each pile. The end-of-file is marked by a test case with $n = 0$ and should not be processed.

Output

For each test case, write a single line with an integer indicating the number of winning moves from the given Nim position.

Sample Input

```
3
7 11 13
2
1000000000 1000000000
0
```

Sample Output

```
3
0
```