IIUC ONLINE CONTEST 2008

Problem A: Da Vinci Code

Input: standard input
Output: standard output

13-3-2-21-1-1-8-5
O, DRACONIAN DEVIL!
OH, LAME SAINT!

The Da Vinci Code is one of the most widely read but controversial books of all time. In this book the writer Dan Brown used a very interesting encryption technique to keep a secret message. It required a great deal of intelligence to decipher the code since there was not enough information available. Here, a similar kind of problem is given with sufficient clues to solve.

In this problem, you will be given a series of numbers, taken from Fibonacci series and a cipher text. Your task is to decipher the text using the decryption technique described below.

Let's follow an example. Any cipher text will consist of two lines. First line is the key which contains some numbers drawn from Fibonacci series. The second line is the actual cipher text.

So, given the following cipher text

```
13 2 89 377 8 3 233 34 144 21 1
OH, LAME SAINT!
```

the output will be:

```
THE MONA LISA
```

For this problem, assume that the first number in Fibonacci series is 1, second one is 2 and each subsequent number is found by adding previous two numbers of the series. So, the Fibonacci series is 1, 2, 3, 5, 8, 13... ...

So, how do we get the string "THE MONA LISA" from the string "OH, LAME SAINT!"? Here some numbers are drawn from Fibonacci series, given in the first line. The first one is **13** which is the **sixth (6^{th})** Fibonacci number in Fibonacci series. So the first uppercase letter in the cipher text **'O'** goes to the **sixth (6^{th})** position in the output string. Second number in the input string is **2** which is the second Fibonacci number and thus **'H'** goes to second position in the output string; then comes **89** which is the **10^{th}** Fibonacci number, so **'L'** which is the **third uppercase letter** in the cipher goes to the **10^{th}** position in the output string. And this process continues until the cipher text ends and hence we find the string "THE MONA LISA". Note that only uppercase letters conveys the message; other characters are simply garbage.

If a Fibonacci number is missing in the input sequence then a blank space is put at its position in the output string. As in the above example **fourth** and **ninth** Fibonacci numbers **5** and **55** are missing. So, two blank spaces are inserted in fourth and ninth positions of the output string. But there must not be any trailing spaces.

Input

Input starts with a line consisting of a single number **T**. **T** test cases follow. Each test case consists of three lines. The first line contains a single positive integer **N**. In the second line there are **N** numbers drawn from the Fibonacci series. The numbers will be separated from each other using spaces. Finally, the third line contains the cipher text to be decrypted.

Output

For each test case, output contains a single line containing the decrypted text. Remember that the decrypted text will contain only uppercase letters.

Constraints

- Value of any input Fibonacci number is less than 2^{31}.
- The length of the cipher text will be at most **100**.

Sample Input	Output for Sample Input
2 11 13 2 89 377 8 3 233 34 144 21 1 OH, LAME SAINT! 15 34 21 13 144 1597 3 987 610 8 5 89 2 377 2584 1 O, DRACONIAN DEVIL!	THE MONA LISA LEONARDO DA VINCI

Problem setter: Mohammed Shamsul Alam
Special thanks: Mohammed Saifur Rahim, Tanveer Ahsan

Problem B: Triples

Input: standard input
Output: standard output

Given a sequence of positive integers. You need to find the number of triples in that sequence. For this problem, (x, y, z) constructs a triple if and only if x + y = z. So, (1, 2, 3) is a triple, where (3, 4, 5) is not.

Input

Each input set starts with a positive integer N. Next few lines contain N positive integers. Input is terminated by EOF.

Output

For each case, print the number of triples in a line.

Constraints

- **$3 \leq N \leq 5000$**

Sample Input	Output for Sample Input
6 1 2 3 4 5 6 6 1 2 4 8 16 32 3 100000000 200000000 100000000 5 1 1 1 2 2	6 0 1 6

Problem setter: Md. Kamruzzaman

Problem C: The 3-Regular Graph

Input: standard input
Output: standard output

The degree of a vertex in a graph is the number of edges adjacent to the vertex. A graph is 3-regular if all of its vertices have degree 3. Given an integer **n**, you are to build a simple undirected 3-regular graph with **n** vertices. If there are multiple solutions, any one will do.

Input

For each test case, the input will be a single integer **n** as described above. End of input will be denoted by a case where **n** = 0. This case should not be processed.

Output

If it is possible to build a simple undirected 3-regular graph with **n** vertices, print a line with an integer **e** which is the number of edges in your graph. Each of the following **e** lines describes an edge of the graph. An edge description contains two integers **a** & **b**, the two endpoints of the edge. Note that the vertices are indexed from **1** to **n.** If it is not possible to build a simple undirected 3-regular graph with **n** vertices, print **Impossible** in a single line.

Constraints

- $1 \le n \le 100$

Sample Input	Output for Sample Input
4 3 0	6 1 2 1 3 1 4 2 3 2 4 3 4 Impossible

Problem setter: Manzurur Rahman Khan
Original idea: Mohammad Mahmudur Rahman

IIUC ONLINE CONTEST 2008

Problem D: GCD LCM

Input: standard input
Output: standard output

The GCD of two positive integers is the largest integer that divides both the integers without any remainder. The LCM of two positive integers is the smallest positive integer that is divisible by both the integers. A positive integer can be the GCD of many pairs of numbers. Similarly, it can be the LCM of many pairs of numbers. In this problem, you will be given two positive integers. You have to output a pair of numbers whose GCD is the first number and LCM is the second number.

Input

The first line of input will consist of a positive integer **T**. **T** denotes the number of cases. Each of the next **T** lines will contain two positive integer, **G** and **L.**

Output

For each case of input, there will be one line of output. It will contain two positive integers **a** and **b**, $a \le b$, which has a GCD of **G** and LCM of **L.** In case there is more than one pair satisfying the condition, output the pair for which **a** is minimized. In case there is no such pair, output -1.

Constraints

- $T \le 100$
- Both **G** and **L** will be less than 2^{31}.

Sample Input	Output for Sample Input
2 1 2 3 4	1 2 -1

Problem setter: Shamim Hafiz

Problem E: The Bus Driver Problem

Input: standard input
Output: standard output

In a city there are **n** bus drivers. Also there are **n** morning bus routes & **n** afternoon bus routes with various lengths. Each driver is assigned one morning route & one evening route. For any driver, if his total route length for a day exceeds **d**, he has to be paid overtime for every hour after the first **d** hours at a flat **r** taka / hour. Your task is to assign one morning route & one evening route to each bus driver so that the total overtime amount that the authority has to pay is minimized.

Input

The first line of each test case has three integers **n**, **d** and **r**, as described above. In the second line, there are **n** space separated integers which are the lengths of the morning routes given in meters. Similarly the third line has **n** space separated integers denoting the evening route lengths. The lengths are positive integers less than or equal to 10000. The end of input is denoted by a case with three 0 s.

Output

For each test case, print the minimum possible overtime amount that the authority must pay.

Constraints

- **$1 \leq n \leq 100$**
- **$1 \leq d \leq 10000$**
- **$1 \leq r \leq 5$**

Sample Input	Output for Sample Input
2 20 5 10 15 10 15 2 20 5 10 10 10 10 0 0 0	50 0

Problem setter: Mohammad Mahmudur Rahman

Problem F: The Sultan's Feast

Input: standard input
Output: standard output

Our beloved sultan is back again, and he is really excited. He will be celebrating his 2nd year anniversary next month. He is arranging a feast for that. But that is where, he is facing some problems.

He is holding a grand feast, and every guest can bring any number of guests. And there guests can also bring their guests and so on. The problem is, he doesn't like everyone, and dislike many of the guests. So, he decided, he won't be inviting every one.

He has assigned a 'like' value to everyone in his guest list. If like value is positive, then he likes them, negative if he dislikes, and zero if neither.

Now, sultan wants to invite his friends in such a way that would maximize the sum of 'like' value of all guests.

Input

First line contains **T**, the number of test cases. Each test case starts with an integer **N**, the number of people in the guest list. The following **N** lines each starts with two integers, l_i and r_i. l_i is the 'like' value of the i-th person, and r_i is the number of people he likes to invite. Rest of the line contains r_i integers, the other guests i-th guest likes to invite. All guests are numbered from 1 to **N**. There is a blank line before each test case.

Output

For each test case, output the case number followed by the maximum 'like' value, or print "Alas, sultan can't invite anyone!" if it's not possible to invite anyone. Sultan can invite, as long as the sum is non negative.

Constraints

- $T \leq 100$
- $N \leq 100$
- $-10000 \leq l_i \leq 10000$, for all i = 1, 2, …, N
- $1 \leq r_i \leq N - 1$
- In the friend list, no person is listed twice, and all numbers in the list are between 1 and N.

Sample Input	Output for Sample Input
3 5 8 2 2 4 -9 1 5 7 1 4 0 1 5 -10 0	Case #1: Alas, sultan can't invite anyone! Case #2: 10 Case #3: 4

4 10 2 2 4 -2 1 4 12 1 4 -10 0 5 10 1 2 -12 1 3 2 1 5 12 1 5 -10 0	

Problem setter: Manzurur Rahman Khan

Problem G: Blobs in the Board

Input: standard input
Output: standard output

You are given a board consists of **R** rows and **C** columns. There are **N** blobs in the board. Each cell of the board might or might not contain a blob. A cell can have at most 8 adjacent cells. A blob can jump over any adjacent blobs to the next empty cell. Thus a blob can jump straight to two squares at any of the 8 directions if the destination cell is empty and the cell between the source cell and the destination cell contains a blob. Blobs cannot jump outside of the board. After the jump the blob in the middle cell of the jump will be removed. The goal is to make jumps carefully so that after N-1 jumps only one blob remains. Your task is to count how many ways the goal can be achieved.

Input

Input consists of several test cases. The first line of the input file contains a single integer **T** indicating the number of test cases. Then **T** test cases follow. Each test case starts with three positive integers **R**, **C** and **N**. Each of the next **N** lines contain two integers **x**, **y** where **i**-th line contains the row position **x** and column position **y** of the **i**-th blob. No two blobs share the same cell.

Output

For each set of input produce one line of output "**Case x: n**", where **x** indicates the test case number (starting from 1) and **n** the number of ways the goal can be achieved.

Constraints

- $R, C \le 4$
- $1 \le x \le R$
- $1 \le y \le C$
- $1 \le i \le N$

Sample Input	Output for Sample Input
3 1 2 1 1 1 3 3 8 2 3 2 1 1 2 3 3 3 2 3 1 1 1 1 3 3 3 3 3 1 2 2 1 2	Case 1: 1 Case 2: 0 Case 3: 2

Problem setter: Istiaque Zaman

IIUC ONLINE CONTEST 2008

Problem H: Binary*3 Type Multiple

Input: standard input
Output: standard output

Mahbub thinks that he has found something interesting but he is not sure whether he is right or not. For each integer he seems to find a multiple of it such that it is only composed of 3s and 0s. Can you help him?

You will be given a positive integer **K**. You have to find a positive multiple of **K** which is only composed of **3**s and **0**s. And in addition to this, the digits of that multiple have to be in non-increasing order. If there are more than one such multiple you have to choose the one which is shortest in length. Even after this if more than one multiple remain then you should choose the lexicographically largest one.

For example,
For **K** = **4**, our desired multiple is **300**.
For **K** = **7**, it is **333333**.
And for **K** = **14**, it will be **3333330**.

Input

The input consists of several lines containing one positive integer **K**.

Output

Your code should produce one line of output containing space separated 3 integers. First integer is the length of the multiple, second integer number of 3s in the multiple and third integer number of 0s in your multiple.

Constraints

- **K** $\leq$ **1000000**

Sample Input	Sample Input
4 7 14	4 7 14

Problem setter: Md. Mahbubul Hasan
Special thanks to: Abduallah Al Mahmud
Source: BdOI Warmup

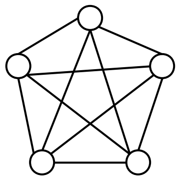
IIUC ONLINE CONTEST 2008

Problem I: Tri-Isomorphism

Input: standard input
Output: standard output

Let *V(G)* be the vertex set of a simple graph & *E(G)* its edge set. An Isomorphism from a simple graph G to a simple graph H is a bijection *f: V(G)→V(H)* such that *uv* ϵ *E(G)* if & only if *f(u)f(v)* ϵ *E(H)*. We say, G is isomorphic to H if there is an isomorphism from G to H.

A complete graph is a simple graph whose vertices are pairwise adjacent: the unlabeled complete graph with n vertices is denoted K_n. For example, the following figure shows K_5.



Finally, a decomposition of a graph is a list of subgraphs such that each edge appears in exactly one subgraph in the list.

Now, given a positive integer **n**, you are to determine if K_n decomposes into three pairwise-isomorphic subgraphs.

Input

First line of each test case consists of a positive integer **n** (n<=100). The end of input will be indicated by a case where n=0. This case should not be processed.

Output

For each test case, print **YES** if K_n can be decomposed into three pairwise-isomorphic subgraphs & **NO** otherwise.

Constraints

- **n < 100**

Sample Input	Output for Sample Input
4 5 0	YES NO

Problem setter: Mohammad Mahmudur Rahman

IIUC ONLINE CONTEST 2008

Problem J: Digit Blocks

Input: standard input
Output: standard output

Often digit blocks are used to teach children formation of numbers. Block sets are available in the market which contains many blocks, each of which has the shape of a digit. Small or large numbers can be formed using them by placing them in a single row. Given the information of the available digit blocks, your job is to find out the total number of different hexadecimal numbers divisible by 5 that can be formed using those blocks. To save you from numbers with leading zeroes you can assume that none of the blocks will have shape of zero. The available blocks may have shapes of ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’,’9’, ‘A’, ‘B’, ‘C’, ‘D’, ‘E’ or ‘F’ which are actually digits of hexadecimal number system.

Input

Input file contains at most 501 lines of inputs. Each line contains a string which contains only digits of hexadecimal number system (but will not contain zero). These digits denote the blocks that are available. For example if the string is “A1BBB5”, then you have two assume that total six blocks are available. Of them three blocks have shape of B, one block has shape of A, another block has shape of 1 and the last one has shape of 5. The string can be at most 16 characters long.

Input is terminated by a line containing a single ‘#’.

Output

For each line of input produce one line of output. This line contains a decimal integer number which denotes the value N. Here N is the number of hexadecimal multiples of 5 that can be formed using the given digits. Note that you can use some or all of the given digits to form number. You can safely assume that N will fit in a 64 bit signed integer.

Sample Input	Output for Sample Input
A1BBB5 B #	4 0

Problem setter: Shahriar Manzoor
Special thanks: Derek Kisman