7th Contest of Newbies

General instructions:

* Unless otherwise specified,

  - The default memory limit is 32 MB.
  - Each input file contains multiple test cases.
  - Each test case starts on a new line.
  - Input is terminated by EOF (end of file).
  - The size of input file is less than 100kB.
  - Double precision is sufficient for floating-point computations.
  - Take $\pi = 2\ cos^{-1}0$.

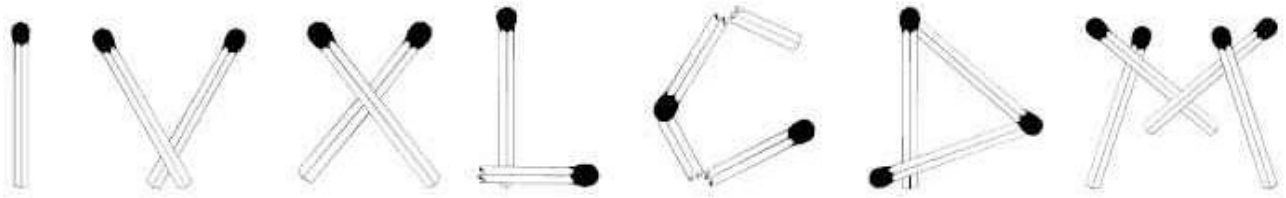
* **Be careful of the data types you use in your codes!!**
* For clarification, please send an email to clarify AT acm.uva.es .
* Clarification replies will be given by email, but they may also be put on the Clarification Board if they are of general interest.
* Hints will be posted on the Clarification Board during the **last hour** of the contest.

The Problemset

Special Thanks to Prof. Miguel Revilla and Shahriar Manzoor for their support and help.

Problem A: Roman Numerals

We would like to build Roman numerals with matches. As you know, Roman numerals are based on the following seven characters: I, V, X, L, C, D, M. Here we introduce the LUSIVERS font, in which the respective characters look like this:



Write a program that counts the number of matches used to build Roman numerals in the LUSIVERS font. This number is exactly the total number of "match heads" in the characters. For instance, to make the number **14** (=XIV), **five** matches are used.

You must follow the "standard modern Roman numerals" (as shown on the Wikipedia page). Expressions like IC or IIII are not allowed.

Input

Input contains multiple lines, each giving a value of N ($1 \leq N \leq 3999$).

Output

For each test case, output the number of matches required to build the number N in Roman numerals.

Sample Input

```
14
2011
```

Sample Output

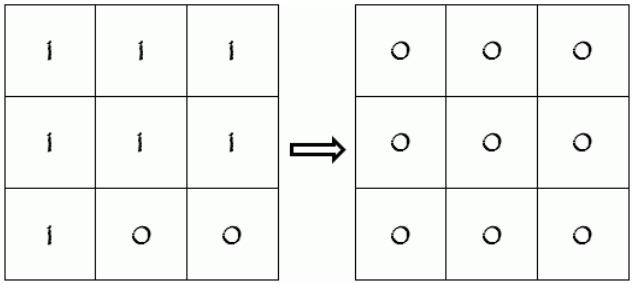
```
5
11
```

Problemsetter: Mak Yan Kei
The LUSIVERS font is available on FontSpace as a freeware

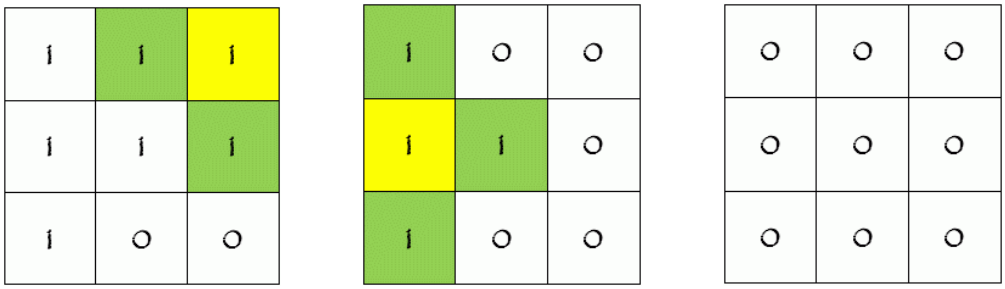
Problem B: NumPuzz I

I have recently downloaded an iPhone app called NumPuzz, developed by Marmottajr. I do not read Portugese, but I find this little game quite suitable for our newbies contest. (The app is available for free at the time this problem description is written.)

You are given a 3-by-3 square matrix with integer entries that may range from 0 to 9, and your task is to make all the entries zeros. You may click on a square and have that entry, together with its neighbouring entries, decreased by one. If an entry to be reduced is a zero, it becomes a 9 after the click. Of course, you aim at solving the puzzle with the fewest clicks possible.



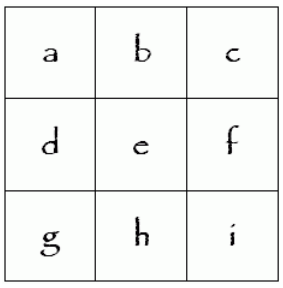
As a simple example, let us suppose that your initial matrix has two zeros in the last two squares, and 1s everywhere else. If you click on the upper right corner, three of the ones turn into zeros. Now it takes just one more click to solve the puzzle. See the illustration below.



A player shows you his sequence of clicks to solve a NumPuzz instance (not necessarily optimal). **Write a program** that returns the initial configuration.

Input

Input contains no more than 100 lines. Each line gives a string of length not exceeding 200, describing the sequence of the player's moves. The meanings of the characters are as follows:



The solution discussed above is described by "**cd**" (click first on c, then on d).

Output

For each test case, print the corresponding matrix in the beginning of the game, using the format shown below.

Notice that the second line in sample input is empty. It means that no clicks are required to solve that puzzle.

Sample Input

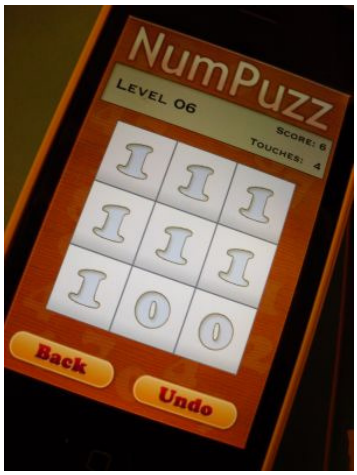
```
cd

cdifbgah
```

Sample Output

```
Case #1:
1 1 1
1 1 1
1 0 0
Case #2:
0 0 0
0 0 0
0 0 0
Case #3:
3 3 3
3 4 3
3 3 3
```

Problemsetter: Mak Yan Kei
NumPuzz is an app by Marmottajr

Problem C: NumPuzz II



We consider again the iPhone game NumPuzz. Please refer to Problem B: NumPuzz I for details of the game.

This task may be thought of as the opposite of Problem B. **Write a program** that accepts as input an initial configuation of a game instance, and returns a solution that takes the fewest clicks possible.

Input

The input format here corresponds exactly to the output format of Problem B. The first line of each case gives the case number, and the following three lines contains the matrix entries at the start of the game.

Output

For each test case, output a string that describes an optimal solution to the given puzzle. Use "**a**" to represent the upper left corner, "**b**" for the square to its right, etc. Print an empty line if no clicks are required. If the puzzle is unsolvable, output "**No solution.**" instead.

Sample Input

```
Case #1:
1 1 1
1 1 1
1 0 0
Case #2:
0 0 0
0 0 0
0 0 0
Case #3:
3 3 3
3 4 3
3 3 3
```

Sample Output

```
cd

cdifbgah
```

Problemsetter: Mak Yan Kei
NumPuzz is an app by Marmottajr

Problem D: 3, 2, 1, 0

This problem relates the digits 3, 2, 1 and 0. Everyone knows that binary numbers are built of 1's and 0's. Now suppose that you are given ***N*** cards of 3 and ***M*** cards of 2. Is it possible to pick ***K*** cards out of them to build a ***K***-digit decimal number such that all the ***K*** least significant digits of its binary representation are zeros?

For example, if you are given one card of each type, you can form the number 32, whose binary representation 100000 ends with ≥ 2 zeros. However, if both cards have the same value, you cannot form a 2-digit number that satisfies the requirement.

Input

Input contains no more than 1000 test cases, each given in a line with three non-negative integers ***N***, ***M*** and ***K***. All input numbers are smaller than 1000.

Output

For each test case, output the **smallest** satisfying ***K***-digit number if found, or "**Impossible.**" otherwise.

Sample Input

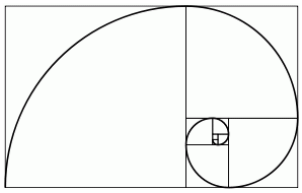
```
1 1 2
2 0 2
```

Sample Output

```
32
Impossible.
```

Problemsetter: Mak Yan Kei

Problem E: Metal Powers



I suppose everyone here has heard of the famous **golden ratio** $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$. This number possesses numerous interesting properties, one of which is illustrated by the following brain-teaser:

"What is the 200th significant digit of φ^{600} (φ to the 600th power)?"

The answer is **9**. This is because $\varphi^n + (1 - \varphi)^n$ is an integer for any positive integer n, and $(1 - \varphi)^{600}$ is an extremely small positive number.

In this problem we introduce two more "metal ratios": the **silver ratio** $\frac{2+\sqrt{8}}{2} \approx 2.414$ and the **bronze ratio** $\frac{3+\sqrt{13}}{2} \approx 3.303$. (These terms are certainly not made up by us; try searching "silver ratio" in MathWorld and you'll see the numbers. Oh wait, Bronze is NOT a metal, hmm...) Curiously, each of these three ratios, when raised to the nth power (for n large enough), is very close to an integer. (Perhaps it is not that curious for those who know *difference equations*.) Let us call these approximate integers "the Metal Powers". Your job is to **write a program** that computes them.

Input

Input has no more than 250 lines, each containing a value of n ($0 \le n \le 100000000$) followed by one of the uppercase letters "G" (for "Golden"), "S" (for "Silver") or "B" (for "Bronze").

Output

For each case, your program should give the corresponding "Metal Power". (For example, the 600th Golden Power means the closest integer to φ^{600}.) If any result contains **more than nine digits**, you only need to give its first three and last three significant digits, together with its total number of digits, as shown in the sample output. You must use the suffixes "th", "st", "nd" and "rd" appropriately.

Sample Input

```
0 G
2 S
27 B
```

Sample Output

```
The 0th Golden Power is 1.
The 2nd Silver Power is 6.
The 27th Bronze Power is 102...036(15 digits).
```

Explanation

The exact value for the third sample case is:

```
102266868132036
```

Problemsetter: Mak Yan Kei
Image taken from Wikimedia Commons

Problem F: Parallel Missions

Ray has always been a fan of **Lineage II**, the (previously) most popular MMORPG (Massively Multi-player Online Role-Playing Game) in Korea and Hong Kong. If you have ever tried the game, you'll know that the procedures of the missions are harsh and time consuming. Most of the time is spent on travelling around the world, to kill a definite amount of a specific kind of monster, or to talk to some NPCs (Non-Player Characters). You may of course avoid the optional missions, but some missions are inevitable.

In order to have stronger combat skills or special abilities (e.g. High grade weapon/armor creation), you need to upgrade to advanced jobs (e.g. from Priest to Bishop) through finishing several extremely tiring missions. One way to minimise the duration is to solve several missions simutaneously. Here's an example:

```
Mission 1 asks player to travel in the following order: 2 → 3 → 5 → 1 → 2
Mission 2 asks player to travel in the following order: 5 → 2 → 3 → 4 → 5
Mission 3 asks player to travel in the following order: 4 → 2 → 4 → 1 → 4
```

Suppose the player is currently in city 4 and assume that the time needed to travel between the cities are the same (say, **1 time unit**). The following route is optimal, taking only 8 time units:

```
4 → 5 → 2 → 3 → 4 → 5 → 1 → 2 → 4
```

It's not always easy to pick an optimal path. You are going to **write a program** to help.

Input

Input contains several test cases. The first line of each case gives a single integer which indicates the number of missions to be solved simutaneously. The first number of the next lines indicate the number of procedures of the missions. The integers following are the cities needed to be travelled (in order). The last line of the case consists of a single integer which locates the starting position of the player.

To make this problem easier we impose the following small input ranges:

- There are no more **10** cities in the world, and at most **10** missions.
- The number of procedures of each mission does not exceed **6**.
- For an optimal path, travelling time required is always less than **20**.

Output

For each test case, output a path such that the travelling time is minimized.

Sample Input

```
3
5 2 3 5 1 2
5 5 2 3 4 5
5 4 2 4 1 4
4
```

Sample Output

```
4 5 2 3 4 5 1 2 4
```

Problemsetter: Raymond Chun