ADA - Análisis y Diseño de Algoritmos, 2019-2
Tarea 1: Semanas 1 y 2

Para entregar el viernes 2 de agosto/domingo 4 de agosto de 2019

Problemas conceptuales a las 16:00 (2 de agosto) en el Departamento de Electrónica y Ciencias de la Computación

Problemas prácticos a las 23:59 (4 de agosto) en la arena de programación

---

Tanto los ejercicios como los problemas deben ser resueltos, pero únicamente las soluciones de los problemas deben ser entregadas. La intención de los ejercicios es entrenarlo para que domine el material del curso; a pesar de que no debe entregar soluciones a los ejercicios, usted es responsable del material cubierto en ellos.

Instrucciones para la entrega

Para esta tarea y todas las tareas futuras, la entrega de soluciones es *individual*. Por favor escriba claramente su nombre, código de estudiante y sección en cada hoja impresa entregada o en cada archivo de código (a modo de comentario). Adicionalmente, agregue la información de fecha y nombres de compañeros con los que colaboró; igualmente cite cualquier fuente de información que utilizó.

¿Cómo describir un algoritmo?

En algunos ejercicios y problemas se pide "dar un algoritmo" para resolver un problema. Una solución debe tomar la forma de un pequeño ensayo (es decir, un par de párrafos). En particular, una solución debe resumir en un párrafo el problema y cuáles son los resultados de la solución. Además, se deben incluir párrafos con la siguiente información:

- una descripción del algoritmo en castellano y, si es útil, pseudo-código;
- por lo menos un diagrama o ejemplo que muestre cómo funciona el algoritmo;
- una demostración de la corrección del algoritmo; y
- un análisis de la complejidad temporal del algoritmo.

Recuerde que su objetivo es comunicar claramente un algoritmo. Las soluciones algorítmicas correctas y descritas *claramente* recibirán alta calificación; soluciones complejas, obtusas o mal presentadas recibirán baja calificación.

Ejercicios

1.1-2 (página 11), 1.2-2, 1.2-3, 1-1 (página 14), 2.1-1, 2.1-2, 2.1-3 (página 22), 2.2-1, 2.2-2 (página 29), 2.3-1 (página 37), 2.3-4, 3.1-1 (página 52), 3.1-4, 3.1-8 (página 53), 3.2-1, 3.2-7, 3-2 columnas O, Ω y Θ (página 61), 3-3 parte *a* (página 62).

Problemas conceptuales

1. Escribir el código de honor del curso.
2. Ejercicios 3 y 5: Orden asintótico y eficiencia algorítmica (Kleinberg & Tardos, páginas 67 y 68).
3. Ejercicio 8: *Safe Rungs* (Kleinberg & Tardos, página 68 y 69).
4. Ejercicio 9: *Pancakes* (Erickson, páginas 49 y 50).
5. Ejercicio 13: *Inversions* (Erickson, página 51).

Problemas prácticos

Hay cinco problemas prácticos cuyos enunciados aparecen a partir de la siguiente página.

A - The ?1?2?...?n = k Problem

Source file name: `problem.py`
Time limit: 1 second

Given the following formula, one can set operators '+' or '−' instead of each '?', in order to obtain a given k:

```
? 1 ? 2 ? . . . ? n = k
```

For example, to obtain $k = 12$, the expression to be used will be:

```
- 1 + 2 + 3 + 4 + 5 + 6 - 7 = 12
```

with $n = 7$.

Input

The first line is the number of test cases, followed by a blank line. Each test case of the input contains an integer k ($0 \leq |k| \leq 1\,000\,000\,000$). Each test case will be separated by a single line.

The input must be read from standard input.

Output

For each test case, your program should print the minimal possible n ($1 \leq n$) to obtain k with the above formula. Print a blank line between the outputs for two consecutive test cases.

The output must be written to standard output.

Sample Input	**Sample Output**
2 12 -3646397	7 2701

B - Helping Fill Bates

Source file name: `bates.py`
Time limit: 1 second

Everyone knows Fill Bates but I guess fewer people know that he was famous in his high school days for a different reason. I just put below the exact lines from one of his short biographies:

> Before high-school graduation, Bates was renowned for designing the class scheduling software that placed him in a class full of girls. With his present success, those girls are probablymutilating themselves for not buying in on an early investment.

Now you are to help him with a completely different purpose. His company has initiated a talent search program in QSA. It has 52 states (the original 50 states plus the two recent troublesome additions Oraq and Malistan). The candidates from the 52 states are given a state identity and a serial number (the serial number is unique). The state identities for the 52 states are the 52 ASCII characters A..Z and a..z. The talent search process is a bit strange. At most 1 million candidates stand in a single line according to the increasing order their serial number (starting with 0 and then 1, 2, 3 etc) with a placard showing only their state identity (after all who wants to hire employees from Oraq and Malistan). Mr. Fill Gates then writes a sequence of characters (only alphabets). If candidates are found in that order of states with increasing serial numbers (some candidates may be skipped for this purpose) then those candidates are taken. Other wise an appropriate message is given.

Input

The input contains only one set of input. First line of the input contains a string S with alphabets only. The length of this string is at most 1 000 000. The next line contains an integer Q ($0 < Q < 3\,501$) which indicates the number of queries. Each of the next Q lines contain one string X of length less than 10 001. These strings are the strings written by Fill Bates.

The input must be read from standard input.

Output

For each query you should output one line. If the candidates X are not found in the order S written by Fill Bates, then you should output a string '`Not matched`' (without the quotes). Otherwise, you should print '`Matched`' (note that an space is printed after '`Matched`') and then the serial of the first candidate in the subsequence and the serial of the last candidate of the subsequence. These two integers should be separated by a single space. If there is more than one such subsequence, then choose the one which has smallest starting serial number. If there is a tie choose the one with the smallest ending serial number.

The output must be written to standard output.

Sample Input	**Sample Output**
aaaaaaaaaaaaaabbbbbbbbbbddddddddddddccccccccccccc 3 aaaaaaaaaaaaaaaaaa aaaaaaaaaaabbbbbbbbbbbc abccc	Not matched Not matched Matched 0 36

C - Steps

Source file name: `steps.py`
Time limit: 1 second

One steps through integer points of the straight line. The length of a step must be nonnegative and can be by one bigger than, equal to, or by one smaller than the length of the previous step.

What is the minimum number of steps in order to get from x to y? The length of the first and the last step must be 1.

Input

Input consists of a line containing n, the number of test cases. For each test case, a line follows with two integers: $0 \leq x \leq y < 2^{31}$.

The input must be read from standard input.

Output

For each test case, print a line giving the minimum number of steps to get from x to y.

The output must be written to standard output.

Sample Input	**Sample Output**
3 45 48 45 49 45 50	3 3 4

D - Divisibility

Source file name: `divisibility.py`
Time limit: 1 second

Consider an arbitrary sequence of integers. One can place + or - operators between integers in the sequence, thus deriving different arithmetical expressions that evaluate to different values. Let us, for example, take the sequence: 17, 5, −21, 15. There are eight possible expressions:

```
17 + 5 + -21 + 15 = 16
17 + 5 + -21 - 15 = -14
17 + 5 - -21 + 15 = 58
17 + 5 - -21 - 15 = 28
17 - 5 + -21 + 15 = 6
17 - 5 + -21 - 15 = -24
17 - 5 - -21 + 15 = 48
17 - 5 - -21 - 15 = 18
```

We call the sequence of integers *divisible by K* if + or - operators can be placed between integers in the sequence in such way that resulting value is divisible by K. In the above example, the sequence is divisible by 7 ($17 + 5 + -21 - 15 = -14$) but is not divisible by 5.

You are to write a program that will determine divisibility of sequence of integers.

Input

The first line of the input contains a integer M indicating the number of cases to be analyzed. Then M test cases follow. For each one of these test cases, the first line contains two integers, N and K ($1 \leq N \leq 10\,000$, $2 \leq K \leq 100$) separated by a space. The second line contains a sequence of N integers separated by spaces. Each integer is not greater than 10 000 by its absolute value.

The input must be read from standard input.

Output

For each case in the input file, write to the output file the word '`Divisible`' if given sequence of integers is divisible by K or '`Not divisible`' if it is not.

The output must be written to standard output.

Sample Input	**Sample Output**
2 4 7 17 5 -21 15 4 5 17 5 -21 15	Divisible Not divisible

E - Number Sequence

Source file name: `sequence.py`
Time limit: 1 second

A single positive integer i is given. Write a program to find the digit located in the position i in the sequence of number groups $S_1S_2\cdots S_k$. Each group S_k consists of a sequence of positive integer numbers ranging from 1 to k, written one after another. For example, the first 80 digits of the sequence are as follows:

```
11212312341234512345612345671234567812345678912345678910123456789101112345678910
```

Input

The first line of the input file contains a single integer t ($1 \leq t$), the number of test cases, followed by one line for each test case. The line for a test case contains the single integer i ($1 \leq i \leq 2147483647$).

The input must be read from standard input.

Output

There should be one output line per test case containing the digit located in the position i.

The output must be written to standard output.

Sample Input	**Sample Output**
2 8 3	2 2