Nivel 2do Año
Problem Set

19 de Octubre de 2018

Problema 1: Pirámide invertida (Inverted pyramid)

SPOJ code: FUCT_FOR_INVPYRA

La entrada es un entero positivo que representa la altura de una pirámida invertida. Imprimir como salida la pirámide invertida.

Por ejemplo, la pirámide invertida de 5 filas es:

```
* * * * * * * * *
  * * * * * * *
    * * * * *
      * * *
        *
```

Entrada:

Contiene la altura de la pirámide invertida.

Restricciones: $0 < n \leq 100$

Salida:

Imprimir la pirámide invertida.

Ejemplos:

Entrada 1

```
5
```

Salida 1

```
* * * * * * * * * *
  * * * * * * * *
    * * * * * *
      * * * *
        * *
          *
```

Entrada 2

```
2
```

Salida 2

```
* * *
  *
```

Problema 2: Divisibilidad (Divisibility)

SPOJ code: SMPDIV

Imprimir todos los enteros a_i tales que a_i es divisible por x pero no es divisible por y, donde $1 < a_i < n < 100000$.

Entrada:

En la primera línea se indica t (t<100), el número de casos de prueba. En cada una de las siguientes t líneas hay 3 enteros: n x y.

Puede asumir también que $x < n$ y que x no es divisible por y.

Salida:

Consiste en t líneas, una para cada caso de prueba. Cada línea contiene los números solicitados en la descripción del problema separados entre sí por un espacio y en orden ascendente.

Ejemplo:

Entrada

```
2
7 2 4
35 5 12
```

Salida

```
2 6
5 10 15 20 25 30
```

Problema 3: Celulares (Cellphones)

SPOJ code: CELLPHONES

Dentro de poco tiempo, Pascual y sus amigos se irán de campamento. El bosque en el que planean acampar es muy hermoso y van a poder realizar muchas actividades al aire libre. Sin embargo, no van a tener energía eléctrica y entonces no podrán cargar sus celulares. Esto es un grave problema, ya que como casi todos hoy en día, los chicos no pueden vivir sin el celular. Para solucionar el problema, Pascual quiere llevar la menor cantidad posible de cargadores portátiles que les permita usar los celulares durante todo el viaje.

Ninguno de los chicos está muy seguro de cómo elegir los cargadores portátiles para no llevar de más, pero ellos saben que sos muy inteligente y por eso te pidieron que los ayudes con este problema. Para facilitarte un poco el trabajo, Pascual armó una lista con los cargadores que tienen disponibles para llevar y además calculó la cantidad de energía eléctrica que van a necesitar para todo el viaje.

Entrada:

La entrada contiene múltiples escenarios de acampe, cada uno representado por 2 líneas consecutivas de texto.

En cada escenario la primera línea contiene dos enteros N ($1 \leq N \leq 2000$) y K ($1 \leq K \leq 10^9$) indicando la cantidad de cargadores que disponen y la cantidad total de energía que necesitarán en el viaje.

La segunda línea incluye N números enteros $c_1 \ldots c_N$ separados por espacios, donde cada c_i ($1 \leq c_i \leq 10^9$) es la capacidad del i-ésimo cargador.

La entrada termina cuando $N = K = 0$.

Salida:

Para cada escenario imprima una línea con el entero que representa la mínima cantidad de cargadores necesaria para que Pascual y todos sus amigos puedan usar sus celulares en el campamento. Si ni siquiera todos los cargadores alcanzan, imprima -1.

Ejemplo:

Entrada

```
5 10
1 2 3 4 5
5 10
1 2 10 4 5
5 6
1 1 1 1 1
0 0
```

Salida

```
3
1
-1
```

Credit for this problem: **Luis Santiago Ré**

Problema 4: Life, Alternate Universe and Everything Else

SPOJ code: ALT_UNIV_TEST

In "The Hitchhiker's Guide to the Galaxy", the amazing first part of the six parts *trilogy* of Douglas Adams' novels, a group of pan-dimensional beings builds a super-computer that works for 7.000.000.000 years to find the Answer to the Ultimate Question of Life, the Universe and Everything Else. That answer turns out to be "42", without any doubt, but the computer properly points out that those beings were never sure about what the proper question really was.

At some point in the third book, one of the main characters is looking for possible questions, and between many options he runs into the following one: "How much is 6 by 9?" Most readers just continue reading without noticing that 6 by 9 is actually 54, and not 42. However, the most detailed and fanatical readers have found out that 6 by 9 actually is 42 if you work in base 13 instead of 10 as usual (in base 10: 6×9=54, but in base 13: 6 and 9 are 6 and 9, but 42 is $4\times13^1 + 2\times13^0$ that is equivalent to 54).

Now that theories of parallel and alternate universes are so trendy, a lot of different answers are arising for a lot of different universes, along with a lot of different questions also in the shape of products; and we want to know, for each alternate universe, if there is a base such that the equality holds for the respective alternates questions and answers. It is known, however, that due to some quantum cosmic limit of the Nature, that the base can never be higher than 16. If that where possible, the law of physics dictate that universe would collapse over itself and would turn into something even more strange.

Input:
The first line contains an integer N with the number of alternate universes to test, and then for each one, a line with 3 numbers: R, P and Q. R is the alternative definitive answer for that universe, and P and Q the terms for the product in the alternative question. The digits in R, P and Q can be the numbers from '0' to '9', and the capital letters from 'A' to 'F' (for bases above 10).

Output:
For each universe, if there is a base lower or equal to 16 in with the product is valid, the output must say "Universe I: B", where I is the number of universe (counting from 1), and B is the smallest base such that P×Q=R in that base; if there is no such base, the output must be "Universe I: Impossible".

Input example:

```
6
42 6 9
42 7 9
54 6 9
40 8 2
125 21 5
24 C 3
```

Output example:

```
Universe 1: 13
Universe 2: Impossible
Universe 3: 10
Universe 4: Impossible
Universe 5: 8
Universe 6: 16
```

Notes

- In the fifth example, the base is 8. Considering the three numbers in that base: $A=2\times8^1+1\times8^0=17$, $B=5\times8^0=5$, $R=1\times8^2+2\times8^1+5\times8^0=85$.
- In the sixth one, the base is 16, and digits 'A', 'B', 'C', 'D', 'E' and 'F' corresponds to 10, 11, 12, 13, 14 and 15 respectively. Then, given that 'C'=12, we get that $P\times Q=12\times3=36$; and $36=2\times16^1+4\times16^0$.

Credit for this problem: **Pablo Novara**

Problema 5: Cifrado circular (Circular encryption) SPOJ code: ENCRYPC

Lucas y su primo desean enviarse mensajes secretos a través de su red social favorita. Desafortunadamente, la red que utilizan no provee encriptación extremo a extremo. Entonces, para enviar en forma secreta los mensajes, deciden codificar cada mensaje cambiando algunos caracteres por otros.

Ellos suelen enviar mensajes de 1000 caracteres como máximo, utilizando únicamente las letras a-z, A-Z, el espacio (cód. ASCII 32) y el punto (cód. ASCII 46).

Lucas propone realizar el cifrado utilizando una secuencia de caracteres S de longitud n, es decir $S = s(1), s(2), ..., s(n)$, donde para todo $i \neq j$ se cumple que $s(i) \neq s(j)$.

Utilizando esta secuencia, la propuesta de Lucas consiste en reemplazar en el contenido de los mensajes cada caracter $s(i)$ por el caracter $s(i+1)$, para todo i en {1, 2, ... $n-1$}, y también en reemplazar $s(n)$ por $s(1)$.

Para hacer más compleja la codificación, su primo propone agregar un número entero k y reemplazar $s(i)$ por s($i+k$), excepto cuando $i+k > n$ en cuyo caso $s(i)$ se reemplaza por $s(i+k-n)$.

Luego de debatirlo acuerdan utilizar la propuesta del primo.

Entrada:

La primera línea de la entrada contiene los números enteros n y k separados por un espacio, y la segunda línea contiene una cadena de n caracteres que representa la secuencia S. La tercera línea contiene un entero m que será el número de mensajes a codificar. Finalmente, las siguientes m líneas incluyen los mensajes a codificar por Lucas o su primo, y cualquier línea adicional es ignorada.

Límites: puede asumir que $2 \leq n \leq 54$, $1 \leq k \leq n-1$, y $1 \leq m \leq 100$.

Salida:

La salida consiste en m líneas de texto, correspondientes a los mensajes codificados según acordaron Lucas y su primo.

Ejemplo:

Entrada

```
17 7
SuPeR EncrYptioN.
2
Please try to solve this problem
It is easy...
```

Salida

```
rlYasYtP.ytPRtsRlvYtPhestu.RblYm
IPtestYasyEEE
```

Credit for this problem: **Pablo Marchetti**

Problema 6: Classroom

SPOJ code: CLASSROOM

You've just walked into your 17th-century quantum cryptography class. The bell hasn't rung yet, but it's close, so there aren't many seats left! You'd like to choose a spot near your friends if possible.

The classroom is represented by an overhead map, in the form of a matrix of integers between -1 and 9 inclusive.

Legend:

- -1 represents an empty seat - you can sit here.
- 0 represents floor space - you can't sit here.
- Any other number between 1 and 9 means a classmate is sitting here, and the numerical value represents how friendly you are with that classmate.

Given class layout, your task is to return the indices of the spot where you should sit to maximize the friendliness of your immediate neighbors (up, down, left, right, or diagonally).
In the event of a tie, you'd prefer to sit closer to the front of the class (ie: choose the option with the lower row index), and if there's still a tie, you'd prefer to be closer to the window on the west wall of the classroom (ie: choose the option with the lower column index).
If there's nowhere for you to sit, print “-1 -1” (without the quotes).

Input format and limits:

The input will be set with several cases. Each case starts with 2 numbers N and M denoting N rows and M columns representing the classroom ($0 \leq N, M < 1000$). A value of N = 0 and M = 0 represents that no more cases will come. After that, N rows containing M numbers will be presented with values from -1 to 9 representing the classroom layout. Consecutive cases are separated by a blank link.

Output format:

For each input case you will have to write the leyend “Case N:X”, where X represents the corresponding test case beginning from 1. The next line should have the row and the column of the selected spot, separated by a single space. Or -1 -1 if there is no posible spot to select. Please, leave a blank line between different solutions.

NOTE: Please, take into account that indexes for rows and columns are meant to start at 0.

Example:

Input:

```
10 10
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 3 6 0 3 -1 0 2 3 0
0 0 0 0 0 0 0 0 0 0
0 8 3 0 8 8 0 9 7 0
0 0 0 0 0 0 0 0 0 0
0 -1 3 0 5 1 0 8 1 0
0 0 0 0 0 0 0 0 0 0
0 2 -1 0 3 4 0 4 5 0
0 0 0 0 0 0 0 0 0 0

1 1
-1

1 1
0

0 0
```

Output:

```
Case N:1
2 5

Case N:2
0 0

Case N:3
-1 -1
```

Explanation of example:

- For Case N:1, the seats available at [2, 5] and [6, 1] would both provide the maximum friendliness value of 3, but since [2, 5] is closer to the front of the class, it's the winner.
- For Case N:2, the only seat available is at 0 0.
- For Case N:3, there is no seat available, so a -1 -1 is printed.

Credit for this problem: **CodeSignal**

Problema 7: ACM (ACronymMaker)

SPOJ code: ACMAKER

The sadists who design problems for ACM programming contests often like to include the abbreviation "ACM" somewhere in their problem descriptions. Thus, in years past, the World Finals has had problems involving "Apartment Construction Management," the "Atheneum of Culture and Movies," the "Association of Cover Manufacturers," "ACM Airlines," the "Association for Computational Marinelife," and even an insect named "Amelia Cheese Mite." However, the number of word combinations beginning with A, C, and M that make sense is finite and the problem writers are starting to run out of ideas (it's hard to think of problems about "Antidisestablishmentarianistic Chthonian Metalinguistics"). Fortunately, modern culture allows more flexibility in designing abbreviations — consider, for example:

```
GDB — Gnu DeBugger
LINUX — either "LINus's UniX" or "LINUs's miniX" or "Linux Is Not UniX"
SNOBOL — StriNg Oriented symBOlic Language
SPITBOL — SPeedy ImplemenTation of snoBOL
```

The rules used in these examples seem to be:

- Insignificant words (such as "of", "a", "the", etc.) are ignored.
- The letters of the abbreviation must appear, in the correct order, as an ordered sublist of the letters in the significant words of the phrase to be abbreviated.
- At least one letter of the abbreviation must come from every significant word (multiple occurrences of a letter are, of course, treated as distinct).

Of course these rules are often broken in real life. For instance, RADAR is an abbreviation for "RAdio Detecting And Ranging". Under our rules (assuming that "and" is an insignificant word), this would not be a valid abbreviation (however, RADR or RADRAN or DODGING would be valid). You have been asked to take a list of insignificant words and a list of abbreviations and phrases and to determine in how many ways each abbreviation can be formed from the corresponding phrase according to the rules above.

Input:

The input file consists of multiple scenarios. Each scenario begins with an integer $100 \geq n \geq 1$ followed by n insignificant words, all in lower case, one per line with no extra white space. (A line containing 0 indicates end of input.) Following this are one or more test cases for this scenario, one per line, followed by a line containing the phrase "LAST CASE". Each line containing a test case begins with an abbreviation (uppercase letters only) followed by a phrase (lowercase letters and spaces only). The abbreviation has length at least 1 and the phrase contains at least one significant word. No input line (including abbreviation, phrase, and spaces) will contain more than 150 characters. Within these limits, however, abbreviations and phrase words may be any length.

Output:

For each test case, output the abbreviation followed by either

is not a valid abbreviation

or

can be formed in i ways

where *i* is the number of different ways in which the letters of the abbreviation may be assigned to the letters in the phrase according to the rules above. The value of *i* will not exceed the range of a 32-bit signed integer.

Example:

Input

```
2
and
of
ACM academy of computer makers
RADAR radio detection and ranging
LAST CASE
2
a
an
APPLY an apple a day
LAST CASE
0
```

Output

```
ACM can be formed in 2 ways
RADAR is not a valid abbreviation
APPLY can be formed in 1 ways
```

Problema 8: Goldbach Graphs

SPOJ code: GOLDG

Christian Goldbach sent a letter to Leonhard Euler in 1742 in which he made the following conjecture:

"Every even number greater than 4 can be written as the sum of two odd prime numbers"

To find the solutions of Goldbach's conjecture for a given even number n ($n > 0$), let us define the directed graph GG(n) (the Goldbach Graph of n) as follows:

Nodes are prime numbers p such that $1 < p < n$.
For each node p there are zero or more outgoing edges, determined by the following rules:
If $p + q = n$ and $q = 1$, then no outgoing edges are related to p.
If $p + q = n$ and $q = p_1\ p_2\ p_3\ ....\ p_k$ is the prime factorization of q (asuming $q > 1$), then for each $i = 1..k$ an edge $p \rightarrow p_i$ is added to graph GG(n). Notice that each p_i must be a prime number. Besides, if $k = 1$ then q is prime and we have a solution to Goldbach's conjecture.

For example:

- GG(2) is empty (it has zero nodes)
- GG(4) has two nodes and one edge.
  nodes = {2, 3}
  edges = {2→2}
- GG(6) has three nodes and three edges
  nodes = {2, 3, 5}
  edges = {2→2, 2→2, 3→3}

  Notice that edge 2→2 appears twice in GG(6) because when $p = 2$ then $q = 4 = 2*2$

Solutions to Goldbach's conjecture are cycles in graph GG(n) of the following types:

- Single-node cycles (Type I): a node p with only one outgoing edge $p \rightarrow p$.
- Double-node cycles (Type II): two nodes p_1 and p_2, such that each one has a unique outgoing edge ($p_1 \rightarrow p_2, p_2 \rightarrow p_1$).

Your task is to inspect the directed graph GG(n) starting from a given node x and searching every node reachable from x for a solution to Goldbach's conjecture. The procedure is successfull if a node belonging to a Type I or Type II cycle is found. In such a case the minimum distance from x to the first node of the cycle found must be reported. Otherwise it should be stated that a solution can not be found.
Your algorithm should take into account that GG(n) can contain other types of cycles besides the ones described here. Otherwise, it can run forever.

Input:

The input contains several lines each one with a different test case. Each line includes a pair of numbers representing the values n and x. You should assume that n is even and also that $2 \leq n \leq 1000$. Although $0 < x < n$ is true, do not assume that x is a valid node of GG(n). The last line of the input contains the number 0 (it is not a test case).

Output:

For each test case output a single line with one of the following:

- Solution found at distance *D*.
- Solution not reachable.
- *x* is not a node!

Where *D* is the minimum distance from *x* to the solution found, as described before.

Example:

Input

```
2 1
4 2
6 2
6 3
12 3
12 11
14 7
20 5
38 11
50 17
540 340
540 31
540 33
0
```

Output

```
1 is not a node!
Solution found at distance 0.
Solution not reachable.
Solution found at distance 0.
Solution not reachable.
Solution not reachable.
Solution found at distance 0.
Solution found at distance 1.
Solution found at distance 2.
Solution found at distance 1.
340 is not a node!
Solution found at distance 0.
33 is not a node!
```

Credit for this problem: **Pablo Marchetti**