Nivel “Libres”
Problem Set

October 19th 2018

Problem A: Life, Alternate Universe and Everything Else

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: $6\times9=54$, 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 1:

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

Output example 1:

```
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: $A=2\times8^1+1\times8^0=17$, $B=5\times8^0=5$, $R=2\times8^2+1\times8^1+5\times8^0=215$, and $215=17\times5$.

* 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**

Problem B: One Week Wonder

A new technology company wants to build a new cell phone and break the market with an unique feature not seen in a while: a battery that can last a whole week with one charge. In order to achieve such an impressive task, the phone will have a very special screen based on electronic ink, a really huge battery, and a very carefully chosen set of applications.

It is known that some apps use much more battery than others. As an example, an instant messenger app has to connect to a server in order to check for new message every minute, even if the screen is off or if the user is working with another app, so it will use a lot of battery; but a note taking application won't do anything while user is not running that app, so won't consume much if the user doesn't take too many notes.

So, the company has made a very exhaustive study analyzing the usage patterns of a lot of potential customers, and built a list with that information that sums up how much battery would each app consume in a week according to its average usage time, and how useful that app actually is in a scale of 10 points. Now is up to you to find the subset of apps from that list that maximizes the points of usefulness while keeping the amount of battery consumed by that subset under 100%.

Input:

First line will contain an integer K that is the number of test cases (K≤100). Each test case starts with an integer N that represents the number of apps available (3≤N≤100), and then N lines containing a word and two integers each. The word is the name of the app (the name will not include spaces), the first integer the percentage of battery that the app consumes in a week (from 1 to 100), and the last one the points of usefulness for that app (from 1 to 10).

Input example 1:

```
1
5
browser 50 5
messenger 40 9
calls 10 2
camera 35 7
notes 20 3
```

Output:

For each test-case, you must print a single line starting with "Case X: U", where X is the number of the test case, starting from 1, and U is the maximum points of usefulness you can get without surpassing 100% of battery usage.

Output example 1:

```
Case 1: 19
```

In the example case, the three selected apps are "camera", "messenger", and "notes". That set adds to 19 points of usefulness (7 from camera, 9 from messenger and 3 from notes), consuming 95% of the battery (35% camera, 40% messenger and 20% notes). There is no other combination that sums up more than 19 points without requiring more than 100% of the battery capacity.

Credit for this problem: **Pablo Novara**

Problem C: Social Networks' Network

Peter is working in the organization of a new programming contest: the Tech-no-Cofee, a contest where programmers have to keep solving problems until they fall asleep; every time someone solves a problem he can drink a cup of coffee, and the last one standing wins. Peter has to advertise the new event in the social networks. But, being the event so new and still unknown, nobody is following the events accounts yet.

So he plans to take advantage of other people's popularity. He has a list of people that he wants to reach, and he also knows somehow who follows who in each social network. So, if he can convince a few very popular people of posting the event's flier, other people will find out about the event when they see these posts due to being following one of those popular users; and these other people will also repost the flier (because the new contest is that awesome) and the many more will see it, and so on.

The question is, how many people he has to convince in order to reach the whole group.

Input:

The first line contains an integer G indicating the number of groups of people (G≤100). For each group, comes first a line with two integers N and K. N represents the number of people in the group (3≤N≤100); and then there are K more lines with pairs of integers. Each pair represents that someone (the first one) follows someone else (the second one) in some social network (it is not relevant which network it is). Within the pair, each person is identified by a number from 1 to N).

Input example 1:

```
2
5 3
2 1
4 3
3 4
8 8
3 1
8 1
7 2
1 4
2 4
3 4
5 4
6 2
```

Output:

For each group, the program must output a line with the format "Group I: 1 person", or "Group I: X people", where I is the group number (counting from 1), and X is the minimum number of people that he has to convince in order to reach everyone in the group.

Output example 1:

```
Group 1: 3 people
Group 2: 1 person
```

Credit for this problem: **Pablo Novara**

Problem D: 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**

Problem E: DNA Sequences

All DNA is composed of a series of nucleotides abbreviated as A, C, G, and T. In research, it can be useful to identify repeated sequences within DNA.

Write a function to find all the 10-letter sequences (substrings) that occur more than once in a DNA molecule s, and return them in lexicographical order. **These sequences can overlap**.

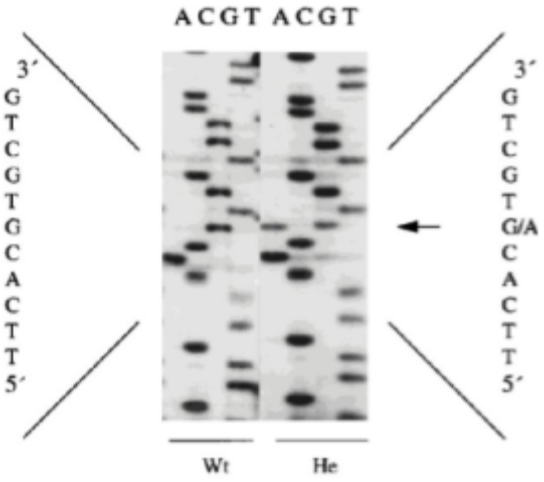


Fig. 1: Autoradiogram showing direct nucleotide sequencing of PCR-ampli®ed human OB DNA of a wildtype (Wt) carrier and in a subject heterozygous (He) for a codon Val110Met variant. The arrow indicates the position of the nucleotide variant.

Input format and limits:
The input will be set with several cases. Each line represent a single case that starts with a string containing uppercase letters corresponding to the nucleotides (A, C, G, T).

Guaranteed constraints:
1 ≤ sequence length ≤ 5000

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 lines should print all of the potential 10-letter sequences that occur more than once in s, in lexicographical order.
Please, leave a blank line between different solutions.

Input example 1:

```
AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT
A
AAAAAAAAAAA
```

OUTPUT:

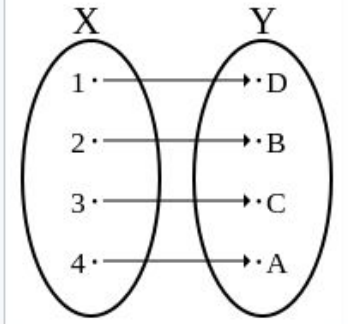
```
Case N:1
AAAAACCCCC
CCCCCAAAAA

Case N:2

Case N:3
AAAAAAAAAA
```

Credit for this problem: **CodeSignal**

Problem F: Top Secret

You are a supersecret agent that specializes on eavesdroping messages of enemies goverments.
This time, you have detected a message coming from a radiofrequency antenna. You have devised the way the messages are encoded, and is as follows: the top secret message contains only uppercase letters from 'A' to 'Z' and has been encoded as numbers using the following mapping system:

'A' -> 1
'B' -> 2
...
'Z' -> 26



A bijective function, $f: X \to Y$, where set X is {1, 2, 3, 4} and set Y is {A, B, C, D}. For example, $f(1) = D$.

Sadly for you, this method is not bijective and for this, there might be more than one way of decode the message. See Fig 1. for an example of a bijective function.
Your task is to determine the total number of ways that the message can be decoded.
Since the answer could be very large, please take it modulo $10^9 + 7$.

Input format and limits:
The input will be set with several cases, one per line. Each case consists in a single string of length of no more that 10^5 characters. You can assume that the string will have valid number digits between 0-9.

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 number of ways of interpreting the message, modulo $10^9 + 7$.
Please, leave a blank line between different solutions.

INPUT:

```
123
```

OUTPUT:

```
3
```

"123" can be decoded as "ABC" (1 2 3), "LC" (12 3) or "AW" (1 23), so the total number of ways modulo ($10^9 + 7$) is 3.

Credit for this problem: **CodeSignal**

Problem G: Eudoxus Numbers

Eudoxus (408-355 BC), as was usual in Greek times, combined philosophy, astronomy and mathematics. They say that he was the first Greek to make a map of the stars. His mathematical work had repercussion in all the world of that time (although it did not have a YouTube channel with thousands of subscribers ... :). Book V of the Elements of Euclid is based on his discoveries about proportions.

At age 23 he attended lectures in Athens, possibly at the Academy of Plato (open c.387 BC)
These days he probably would have followed several online courses, and Plato would have had a YouTube channel with thousands of followers, but we better not get distracted ...

The Eudoxus numbers are defined as follows:

$X_r = Y_r + Y_{r-1}$ if r >= 1
$Y_r = X_{r-1} + Y_{r-1}$ if r >=1
$X_0 = 1$
$Y_0 = 0$

Input:
The first line in the input contains an integer T (1 <= T <= 10), with the number of test cases. Then for each one, comes a line with one integer number N (1 <= N <= 25).

Output:
For each integer number N, you must display in each line, the X_n and Y_n values of Eudoxus numbers, separated by a space character.

Input example 1:

```
3
2
4
5
```

Output example 1:

```
3 2
17 12
41 29
```

Credit for this problem: **"Ejercicios creativos y recreativos en C++" (Pareja, Palao, Gregorio, Llana, Martínez)**

Problem H: Largest Palindrome Product

A palindromic number reads the same both ways. All numbers in base 10 (and indeed in any base) with one digit are palindromic. The number of palindromic numbers with two digits is 9: {11, 22, 33, 44, 55, 66, 77, 88, 99}.

Do you know that in 2018, a paper was published demonstrating that every positive integer can be written as the sum of three palindromic numbers in every number system with base 5 or greater ? (this is a great discover thinking in a problem for TecnoMate 2019!)

In some countries, palindromic numbers are considered a good fortune signal, and people collects things with these numbers, for example bus tickets...

The smallest 6 digit palindrome made from the product of two 3-digit numbers is 101101 = 707 x 143.

Find the largest palindrome made from the product of two 3-digit numbers which is less than N .

Constraints:

- 1 <= T <= 100
- 101101 < N < 1000000

Input:

The first line in the input contains a value *T*, integer (1 <= T <= 10), with the number of test cases. This is followed by *T* lines, each containing an integer *N* .

Output:

Print the required answer for each test case in a new line.

Input example 1:

```
2
101110
800000
```

Output example 1:

```
101101
793397
```

Credit for this problem: **Project Euler**

Problem I: Goldbach Graphs

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 $p->p_i$ 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 $p->p$.
- Double-node cycles (Type II): two nodes p_1 and p_2, such that each one has a unique outgoing edge ($p_1->p_2, p_2->p_1$).

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 $2 <= n <= 1000$. 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).

Input example 1:

```
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:

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.

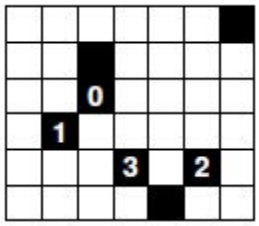
Output example 1:

```
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!
```

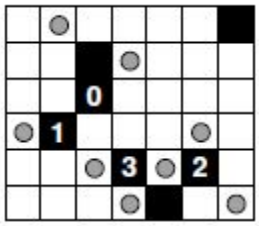
Credit for this problem: **Pablo Marchetti**

Problem J: Light Up

Light Up is a puzzle set in a rectangular board divided in smaller squares. Some squares in the board are "empty" (white squares the figure below), some squares are "barriers" (dark squares in the figure below). A barrier square may have an integer number i associated to it ($0 \le i \le 4$).



(a)



(b)

Figure: (a) Puzzle with 6 rows, 7 columns and 7 barriers; (b) a solution to the puzzle.

In this puzzle the goal is to "light up" all the empty squares by placing lamps in some of them (lamps are depicted as circles in the figure). Each lamp illuminates the square it is on, plus all squares in line with it, horizontally or vertically, up to a barrier square or the board end.
A winning configuration satisfies the following conditions:

- all empty squares must be lit;
- no lamp may be lit by another lamp;
- all numbered barrier squares must have exactly that number of lamps adjacent to them (in the four squares above, below, and to the side);
- non-numbered barrier squares may have any number of lamps adjacent to them.

You must write a program to determine the smallest number of lamps that are needed to reach a winning configuration.

Input:

The input contains several test cases. The first line of a test case contains two integers **N**, **M** indicating respectively the number of rows and the number of columns of the board ($1 \le \mathbf{N} \le 7$, $1 \le \mathbf{M} \le 7$). The second line contains one integer **B** indicating the number of barrier squares ($0 \le \mathbf{B} \le \mathbf{N} \times \mathbf{M}$). Each of the next **B** lines describe a barrier, containing three integers **R**, **C** and **K**, representing respectively the row number ($1 \le \mathbf{R} \le \mathbf{N}$), the column number ($1 \le \mathbf{C} \le \mathbf{M}$) and the barrier number ($-1 \le \mathbf{K} \le 4$); $\mathbf{K} = -1$ means the barrier is unnumbered. The end of input is indicated by $\mathbf{N} = \mathbf{M} = 0$.

Output:

For each test case in the input your program must produce one line of output, containing either an integer indicating the smallest number of lamps needed to reach a winning configuration, in case such a configuration exists, or the words 'No solution'.

Input example 1:

```
2 2
0
2 2
1
2 2 1
6 7
7
2 3 -1
3 3 0
4 2 1
5 4 3
5 6 2
1 7 -1
6 5 -1
0 0
```

Output example 1:

```
2
No solution
8
```

Credit for this problem: **Ricardo Anido**