DevsuCodeJam 2014

Piece of cake (2 or 3 points each one)

1. Compress - 2 points

Implement a method that receives a string that doesn't contain numbers, and performs a very basic string compression.

The compression replaces adjacent repeated characters for the character and their count. For example: "aaaabbdxy" would become "a4b2dxy". Spaces and any other character can also be compressed, of course.

When an empty string is received, it should return an empty string. If null is received, return null.

2. Encode - 2 points

Implement a method that reverses a string, with all the first characters of each word capitalized, and all other letters in lowercase. The words should always be splitted in a space.

For example, if the method receives "Hello my friend", it should return "Dneirf Ym Olleh".

If the method receives null, it must return null. If an empty string is received, it should return the same string.

Pretty easy, right? Before coding, read the rules! :) **RULES**: You cannot use ANY "reverse" or "split" method provided by the language.

3. Find and Sum - 3 points

Just find the numbers inside a string, and return their sum. Digits that are together represent just one number.

All the numbers must be considered integers. they should be considered as different numbers if they are separated by any character different than a digit, even if it's a comma or point.

For example, given: "Genius is 1 percent inspiration, 99 percent perspiration", the method should return 100, since 1 + 99 = 100.

And given: "When I was 18, I was in 1st year, and had 25 classmates", it should return 44, since 18+1+25 = 44.

4. Simple cipher - 3 points

Your goal on this exercise is to create a method that will encrypt the received string.

The encryption method is pretty simple:
Characters that are on an odd position should be replaced with the next character in the alphabet, while characters in an even position, should be replaced with the previous character in the alphabet. In case of Z, it should be replaced by A, and vice versa. Only [a-z] and [A-Z] characters should be transformed (Do not transform or consider special characters).

The **1**st character is **odd**, the **2**nd character is **even**, and so on.

For example:

`The quick brown fox jumps over the lazy dog 12345 :) * zzzZZZAAAaaa`
should be transformed to:
`Ugf rtjbl cqpvo gny ktnot pufq sid kbyz enh 12345 :) * yayAYAZBZbzb`

5. **A simple series exercise - 3 points**

Write a method that receives a positive integer, and return the n element of this series:

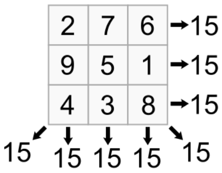
2, 5, 10, 17, 26, ...

If it receives **1**, it should return the first element (2). It it receives 2, it should return the next element (5), and so on. If the method receives zero or a negative integer, it should return -1.

Easy (4 to 5 points each one)

6. **Magic Square - 4 points**

From Wikipedia: “a magic square is an arrangement of distinct numbers (i.e. each number is used once), usually integers, in a square grid, where the numbers in each row, and in each column, and the numbers in the main and secondary diagonals, all add up to the same number.”



2	7	6
9	5	1
4	3	8

Create a method that receives a matrix of integers, and returns true if it’s a magic square, and false otherwise. Pretty easy, right?

If the matrix is not a square, it should return false.

7. **Special Merge - 5 points**

Implement a method that receives two unsorted array of integers, and merges them into one sorted array. Return the sorted array of integers. (The array must be sorted in ascending order, from lowest to highest number).

The method shouldn’t fail even if one or two of the parameters are null, or if they are empty arrays (If both are null, or if both are empty, it will return null).

RULES: You cannot use ANY sorting method provided by your language (eg. array.sort(), sort(), etc). You have to implement your own sorting method if needed. Everything has to be done using arrays and primitive variables. (You cannot use any objects like List, ArrayList, Map, Vector, etc.)

Medium (6 to 8 points each)

8. **Mix Names - 7 points**

Given two arrays, one of first names, and one of last names, generate an array of all the full names that could be generated with two first names, and two last names, considering that the first name cannot be repeated in a full name, whereas the last name can.

For example, given the following arrays: [Juan, Pedro] and [Perez, Lara]
The method should return an array with the following strings:

Juan Pedro Perez Perez
Juan Pedro Perez Lara
Juan Pedro Lara Perez
Juan Pedro Lara Lara
Pedro Juan Perez Perez
Pedro Juan Perez Lara
Pedro Juan Lara Perez
Pedro Juan Lara Lara

The full names can be in any order in the result array, but you have to return all the possible combinations. The received arrays can have any length. If no names can be generated, an empty array should be returned.

9. Longest Substring - 8 points

Create a method that receives two strings, and return the longest substring in the first string, that doesn't contain any character of the second string.

10. Another Series Exercise - 8 points

Write a method that receives a positive integer, and return the corresponding element of this series:

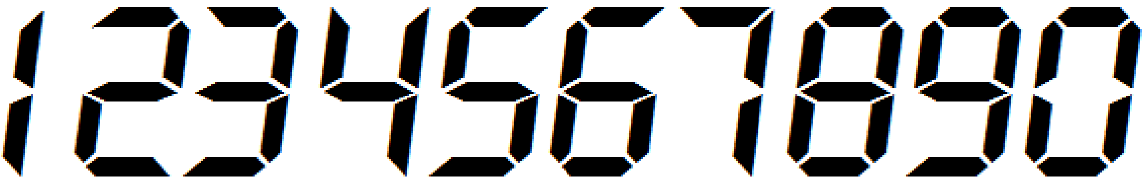
3, 4, 4, 12, 6, 20, 9, 28, 13, 36, 18, 44, ...

If it's **1**, it should return the first element (3). It it receives **2**, it should return the next element (4), and so on. If the method receives zero or a negative integer, it should return -1.

Difficult (10 to 12 points Each)

11. Draw numbers - 10 points

Implement a method that receives an integer from 0 to 99, and return a matrix of characters values with the numbers drawn on them, using asterisks, simulating a digital display that displays the following numbers:



The width of each number should be 3 asterisks, and the height should be 5. Empty spaces should be filled with spaces. There must exist one empty column between the two numbers.

For example:
For number 12, the matrix returned should be:

```
[[ , ,*, ,*,*,*],
 [ , ,*, , , ,*],
 [ , ,*, ,*,*,*],
 [ , ,*, ,*, , ],
 [ , ,*, ,*,*,*]]
```

In other words, if the method receives 12, the matrix returned must be:

```
matrix[0][0] = ' ';    matrix[1][0] = ' ';    matrix[2][0] = ' ';    matrix[3][0] = ' ';    matrix[4][0] = ' ';
matrix[0][1] = ' ';    matrix[1][1] = ' ';    matrix[2][1] = ' ';    matrix[3][1] = ' ';    matrix[4][1] = ' ';
matrix[0][2] = '*';    matrix[1][2] = '*';    matrix[2][2] = '*';    matrix[3][2] = '*';    matrix[4][2] = '*';
matrix[0][3] = ' ';    matrix[1][3] = ' ';    matrix[2][3] = ' ';    matrix[3][3] = ' ';    matrix[4][3] = ' ';
matrix[0][4] = '*';    matrix[1][4] = ' ';    matrix[2][4] = '*';    matrix[3][4] = '*';    matrix[4][4] = '*';
matrix[0][5] = '*';    matrix[1][5] = ' ';    matrix[2][5] = '*';    matrix[3][5] = ' ';    matrix[4][5] = '*';
matrix[0][6] = '*';    matrix[1][6] = '*';    matrix[2][6] = '*';    matrix[3][6] = ' ';    matrix[4][6] = '*';
```

The matrix has always to be a 5 x 7 matrix. If the number received is lower than 10, the first space should be filled with a zero.

For example, for number 6, the matrix returned should be:

```
[[*,*,*, ,*,*,*],
 [*, ,*, ,*, , ],
 [*, ,*, ,*,*,*],
 [*, ,*, ,*, ,*],
 [*,*,*, ,*,*,*]]
```

If the number received is lower than 0 or higher than 99, return a matrix with “ER” drawn, as follows:

```
[[*,*,*, ,*,*,*],
 [*, , , ,*, ,*],
 [*,*,*, ,*,*,*],
 [*, , , ,*,*, ],
 [*,*,*, ,*, ,*]]
```

12. Justify - 12 points

Implement a method that receives two parameters, one string, and one positive integer that will be the line width. Return the text **justified**, considering the line width.

For example, given the string:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute
irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
deserunt mollit anim id est laborum.
```

and line width 50, it should return:

```
Lorem ipsum dolor sit amet, consectetur adipiscing
elit,  sed  do eiusmod tempor incididunt ut labore
et  dolore magna aliqua.  Ut enim ad minim veniam,
quis  nostrud exercitation ullamco laboris nisi ut
aliquip  ex  ea commodo consequat. Duis aute irure
dolor  in  reprehenderit  in  voluptate velit esse
cillum  dolore eu fugiat nulla pariatur. Excepteur
sint  occaecat  cupidatat  non  proident,  sunt in
culpa  qui  officia  deserunt  mollit  anim id est
laborum.
```

Rules:
The line width must always be the same.
The words cannot be splitted. The separation must always be done in a space.
The lines must not begin or end with a space.
To complete the line width, the spaces should be equally distributed, starting from the first space to the last, for example, in the second line of the example, we have:

```
elit,  sed  do eiusmod tempor incididunt ut labore
     ^^   ^^  ^       ^      ^          ^  ^
```

and in the 9th line:

```
culpa  qui  officia  deserunt  mollit  anim id est
     ^^   ^^       ^^        ^^      ^^    ^  ^
```

The last line must not be justified.
The separation of each line must be done with the character “\n”.

If the string is null, or the linewidth is zero or negative, throw an Exception (if language is Java, C#), or return -1 (if the language is PHP, JS).

13. Draw an Octagon - 11 points

Write a function that receives an integer value, and returns a matrix of characters filled with an octagon of asterisks with the given value as the size of each side. The elements that are not asterisk should be filled with spaces.

n can be any number between 2 and 255. For n<2 or n>255, the function should throw an Exception (if language is Java or C#), or return -1 (if language is PHP or JS).

Example:

By calling your_function(3), your function should return a matrix that can be used to print an octagon like this:

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

In other words, your function should return a matrix with these values:

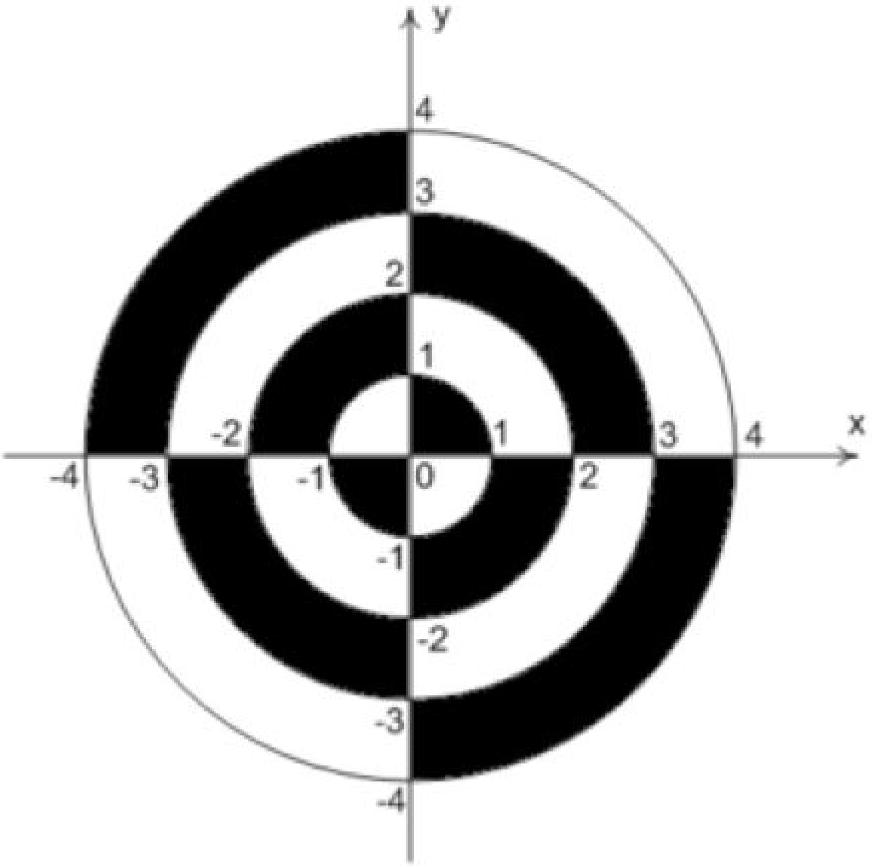
```
a[0][0] = ' ';      a[2][0] = '*';      a[4][0] = '*';      a[6][0] = ' ';
a[0][1] = ' ';      a[2][1] = ' ';      a[4][1] = ' ';      a[6][1] = ' ';
```

```
a[0][2] = '*';        a[2][2] = ' ';        a[4][2] = ' ';        a[6][2] = '*';
a[0][3] = '*';        a[2][3] = ' ';        a[4][3] = ' ';        a[6][3] = '*';
a[0][4] = '*';        a[2][4] = ' ';        a[4][4] = ' ';        a[6][4] = '*';
a[0][5] = ' ';        a[2][5] = ' ';        a[4][5] = ' ';        a[6][5] = ' ';
a[0][6] = ' ';        a[2][6] = '*';        a[4][6] = '*';        a[6][6] = ' ';

a[1][0] = ' ';        a[3][0] = '*';        a[5][0] = ' ';
a[1][1] = '*';        a[3][1] = ' ';        a[5][1] = '*';
a[1][2] = ' ';        a[3][2] = ' ';        a[5][2] = ' ';
a[1][3] = ' ';        a[3][3] = ' ';        a[5][3] = ' ';
a[1][4] = ' ';        a[3][4] = ' ';        a[5][4] = ' ';
a[1][5] = '*';        a[3][5] = ' ';        a[5][5] = '*';
a[1][6] = ' ';        a[3][6] = '*';        a[5][6] = ' ';
```

14. Find the color - 11 points

Given the following graphic:



The objective of this exercise is to create a method that receives two integers, x and y, and return “white” or “black”, depending on the color of the graphic. Both **x** and **y** can be any integer, positive or negative.

IMPORTANT RULE: All the points located in the border of the areas are black.

For example:

When the input is -2 and 1, it should return “white”.

When the input is 2 and 1, it should return “black”.

When the input is 4 and 3, it should return “black”.

When the input is 4 and -3, it should return “black”.

15. Wireworld - 11 points

Wireworld is a game that simulates electronic circuits, using cells that represent electrons traveling across conductors.

The rules of the game are:[1]

“A Wireworld cell can be in one of four different states, usually numbered 0–3 in software, modeled by colors in the examples here:

0: Empty (Black)

1: Electron head (Blue)

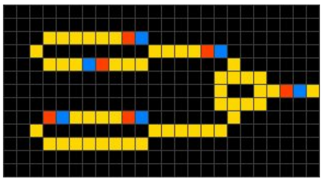
2: Electron tail (Red)

3: Conductor (Yellow)

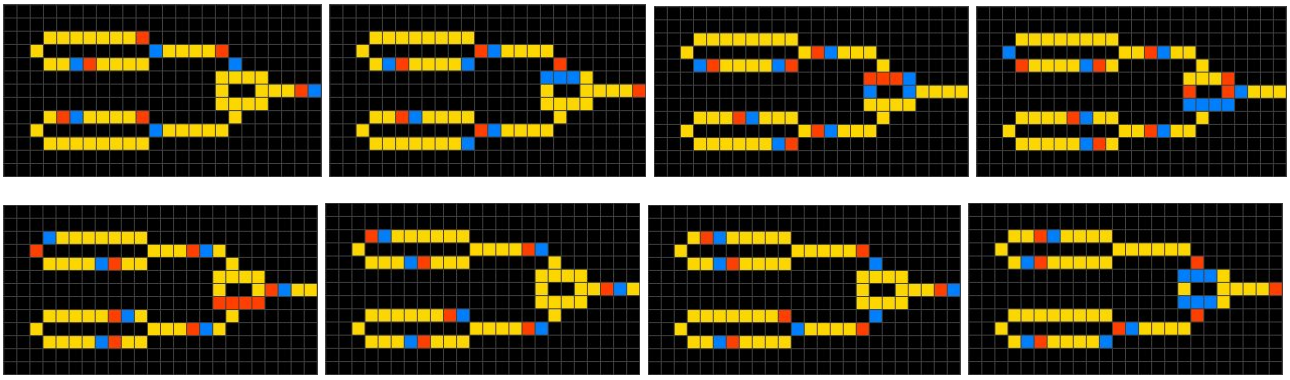
As in all cellular automata, time proceeds in discrete steps called generations (sometimes "gens" or "ticks"). Cells behave as follows:

- Empty → Empty
- Electron head → Electron tail
- Electron tail → Conductor
- Conductor → Electron head if exactly one or two of the neighbouring cells are electron heads, or remains Conductor otherwise.”

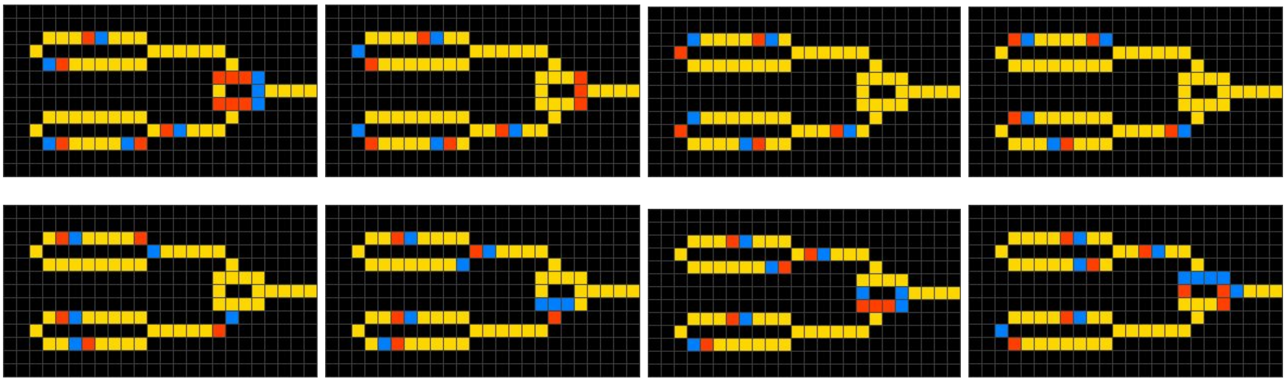
For example, given this initial state:



The following graphics should the evolution of the matrix through 16 generations.



[1] From Wikipedia



Create a method that receives two parameters: A matrix with the initial state (Generation 0), and a positive integer which represents the generation that should be returned.

If the generation is zero or lower, return null;

Bonus

16. **Wireworld Bonus - 8 points**

Create a method similar to the previous exercise, but that also allows negative numbers and zero.

If zero is received, the same matrix must be returned.

If a negative number is received, return the matrix of the corresponding old generation. Consider using the graphics as a reference.

It should continue working for positive numbers as well of course.