

DevsuCodeJam 2019

Piece of cake (5 to 15 points each)

1. Encrypt your message - 5 points

You have been tasked with developing a new way to encrypt communications. Basically, every vowel of the input message will have to be preceded by another string, called the key. The function will receive **two string** parameters: the first one will be the key, and the second one, the message. The function will return a **string**.

Example:

Key: dcj

Message: I love prOgrAmming!

This should return dcjI ldcjovdcje prdcjOgrdcjAmmdcjing!

The letter y is **NOT** going to be considered a vowel and vowels that contain accents are not taken into consideration.

When an empty string is received, it should return an empty string. If message is null or empty, return an empty string. If key is null or empty, then use DCJ as the default.

2. Track my time - 10 points

Given an array of integers representing a time in ms, calculate the sum of all those values and present them in another integer array where the first element has the number of days, the second element has the number of hours, the third element has the number of minutes, the fourth element has the number of seconds and the fifth element has the number of milliseconds. If a time is negative, it counts as 0 ms. If input is null or empty, it counts as 0 ms.

Example:

Input: [65647440, 199644521]

Output: [3, 1, 41, 31, 961]

3. Nth case - 10 points

Write a method that receives an **integer** and a **string**, and returns the exact same string where every k^{th} ($k > 0$) char has had its case changed.

If null is received as the message, return an empty string.

If $n \leq 0$, return the same message.

Example:

Input: (3, Greetings, this is AN EXAMPLE!)

Output: GrEetIngS, ThiS iS An ExAMpLE!

4. Series reloaded - 15 points

Write a method that receives a **positive integer**, and returns the n^{th} element of this series.

-3, -2, 1, 6, 13, 22, 33, 46, 61, 78 ...

If it receives 1, it should return -3. If it receives 2, it should return the next element -2, and so on. If $n \leq 0$, it should return -1.

Easy (20 to 25 points each one)

5. Decrypt my message - 20 points

You have been tasked with decrypting some very cool encrypted messages. Basically, *most* vowels (not necessarily all of them) of the input message have been preceded by another string, called the key.

The function will receive **two string** parameters: the first one will be the key, and the second one, the encrypted message. The function will return a **string**

Example:

Key: hi

Message: hhiighhiEr hiordhiEr fhiUncthiihions

This should return highEr ordEr fUnctions

Please note that if the key is **NOT** followed by a vowel, then it must be treated as part of the message.

Example:

Key: ll

Message: hllalls

This should return halls.

The letter y is **NOT** going to be considered a vowel and vowels that contain accents are not taken into consideration.

When null or an empty string are received, it should return an empty string. If key is null or empty, then use DCJ as default.

6. Clock Angles - 20 points



Given an **array of strings** containing string representations of time in hh:mm pattern, where hh is an integer between 00 and 23 and mm is an integer between 00 and 59, calculate the sum of the angles measured between the clock hands. If the input is not on the hh:mm pattern, you need to subtract 100 from the result. The angle **MUST** be the one between the hour hand and the minute hand, clockwise, and not the other way around. We ignore the movement caused by seconds. Every angle must be between 0 and 2π (between 0 and 360°).

Example:

Input: ["12:00", "17:30", "blabla", "20:21", "26:88"]

Output: 50.5

The function will receive a **string[]**, and it must return a **floating point number**.

Medium (30 to 40 points each)

7. Eerie Mob - 30 points

We all love emojis. ASCII emojis have existed for a while now, including this guy: (-_-). You can represent a **mob** of these like this: (-(-_(-_-(-_(-_-)_-)_-)_-)_-)_-). **Looks scary!** That mob has 11 guys!

The rules to create a mob are as follows:

1. A mob can have **between 1 and 255 guys, 255 included**.
2. If there's an **even** number of guys, then there should be one more on the left than on the right: **eg 4:** (-_(-_(-_-)_-)_-)
3. There are four different types of guys: a complete guy (-_-), a side guy _-), a partial guy -_-) and a final guy -).
 - a. The complete guy must be in the middle, and there must only be one per mob.
 - b. A final guy must be on both sides, as long as there are **strictly more than 7 guys**.
 - c. Let's say the **complete guy** is at the k^{th} position. Every $(k \pm 3n)^{\text{th}}$ guy must be a partial guy.
 - d. All the rest are side guys.

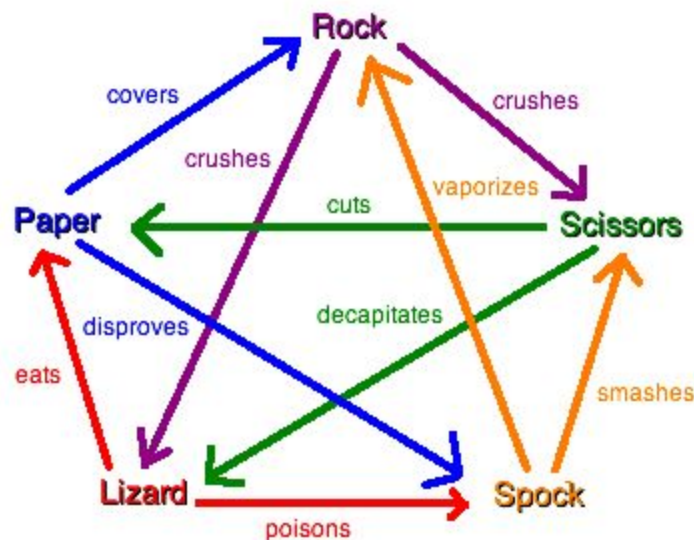
An example, a mob that has 14 guys is this one.

(-(-_-(-_(-_-(-_(-_-)_-)_-)_-)_-)_-)_-)

Create a function that receives an integer, indicating how many guys are on your mob and returns a string with the resulting mob. If the parameter does not comply with the first rule, or if it's null or empty, then the function should return (0_o)

8. Rock Paper Scissors Lizard Spock¹ - 25 points

Rock Paper Scissors Lizard Spock is an extension of the original Rock Paper Scissors (Rochambeau). It adds two possibilities, them being a lizard and Star Trek™'s Spock. Its rules are as follows:



Two friends are playing and you are in charge of writing down who won.

Implement a function that takes **two string[]** containing any of the following words: rock, paper, scissors, lizard or spock.

The first array represents what player 1 drew while playing, while the second array represents what player 2 drew. If one of the corresponding strings is different than any of the given options, that player loses the round. If both strings are different than any of the given options, no points are awarded that round.

If for some reason the two string arrays are not of the same size, then you should take the smallest of the sizes and continue comparing as normal. This also means that if any of the inputs is null or empty, then both players tied.

The function should return one of five case-sensitive **strings**:

¹ Game invented by Sam Kass and Karen Bryla..

- players tied, when both players tied
- player 1 won by 1 point, when player 1 wins by one point
- player 1 won by X points, when player 1 wins by X points
- player 2 won by 1 point, when player 2 wins by one point
- player 2 won by X points, when player 2 wins by X points

Input: ["rock", "lizard ", "scissors", "scissors"] ["paper", "rock", "spock", "paper"]

Output: player 2 won by 2 points

9. Minesweeper map - 30 points

Minesweeper is a known computer puzzle game that debuted in the 1960s consisting of a grid that could contain a mine. You have to build a minesweeper solver, knowing the bomb placements. You are presented with an **M x N** matrix, containing either an **empty string or whitespace** or an **x**, which represents a bomb.

You need to fill the matrix with numbers: every cell has a number representing how many adjacent squares contain mines. If no mines are adjacent, then you must fill the cell with a 0.

Example:

Input:

```
[ [ , , ,x, , , ,x, , ],
  [ , ,x, , ,x, , , ,x],
  [ ,x, , , ,x,x,x,x, ],
  [ , , ,x, ,x, ,x, , ],
  [ , , , , ,x,x,x, ,x],
  [ ,x, , ,x,x,x, , , ] ]
```

Output:

```
[
  [0,1,2,x,2,1,2,x,2,1],
  [1,2,x,2,3,x,5,4,4,x],
  [1,x,3,2,4,x,x,x,x,2],
  [1,1,2,x,4,x,8,x,5,2],
  [1,1,2,2,5,x,x,x,3,x],
  [1,x,1,1,x,x,x,3,2,1]
]
```

You have to implement a function that takes a **char[][]** and returns another **char[][]**. No input will be null.

10. Product subarray - 35 points

Given an **array of floating point numbers**, find the contiguous subarray with the largest product.

The subarray can be of any length, it could even be the whole array. The input will never be null.

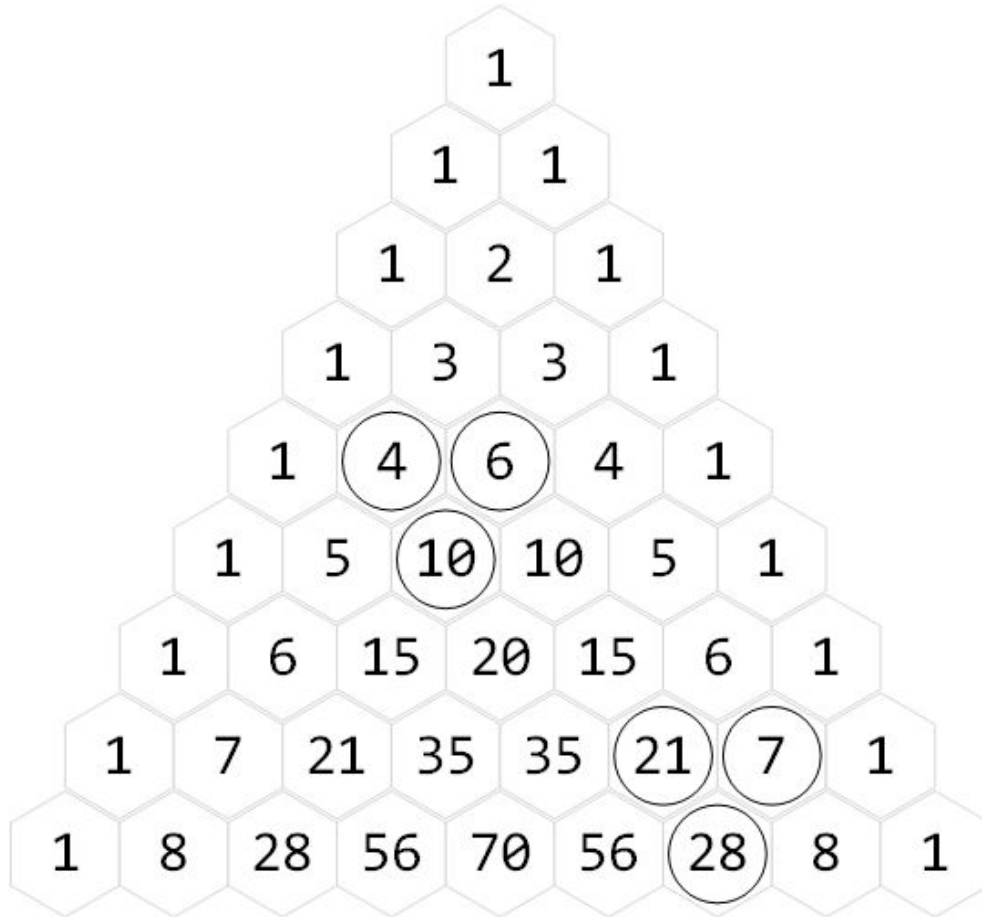
Example:

Input: [-3.2, 4.2, 7, 5.4, -2.2, -2.5]

Output: [-3.2, 4.2, 7, 5.4, -2.2]

11. Pascal Triangle - 35 points

Pascal's triangle is a triangular array that contains all of the binomial coefficients. The triangle may be constructed in the following manner: In row 0 (the topmost row), there is a unique nonzero entry 1. Each entry of each subsequent row is constructed by adding the number above and to the left with the number above and to the right, treating blank entries as 0. For example, the number 10 (on the 5th row) is obtained by adding 4 and 6 on the 4th row. Similarly, 28, on the 8th row is obtained by adding 21 and 7, from the 7th row.



Implement a function that takes two integer parameters, x and y , and returns the $(x,y)^{\text{th}}$ element of the Pascal triangle, 0-based..If the $(x,y)^{\text{th}}$ element does not exist, return -1.

Example:

Input: 2, 1

Output: 2

Input: 6, 4

Output: 15

12. Distinct characters - 35 points

Given a string, find the length of the smallest substring that contains every distinct character in the string. Of course, characters may appear more than once in that substring. The operation must not be case sensitive, meaning **A and a will correspond to the same character**. If the message is null, return 0;

Example:

Input: gamer programming

Output: 13

The output corresponds to the substring `er programmin`.

Difficult (50 to 60 points Each)

13. Toeplitz matrix - 40 points

In linear algebra, a Toeplitz matrix is an $M \times N$ matrix in which all of the diagonals from top left to bottom right have the same value. You must make a function that takes an integer matrix as the input, and returns an integer, indicating how many distinct elements there are. If the matrix is **NOT** a Toeplitz matrix, return -1. Remember that single elements count as a diagonal. No input will be null.

Example:

Input:

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

Output: 7

14. Sum to zero - 35 points

Given an **integer array**, remove all consecutive nodes that sum to zero, and return the remaining nodes. An empty array can very well be an output. If null is received, return an empty array.

Example:

Input: [3, 4, -7, 5, -6, 2, 5, -1, 8]

Output: [5, 8]

15. Kaprekar's Constant - 35 points

The number 6174 is known as Kaprekar's constant. This mathematician discovered a very neat property of this number: for all four-digit numbers with at least two distinct digits, repeatedly applying an algorithm eventually results in this value. The procedure is as follows:

1. For any 4 digit number with at least two distinct digits, create two numbers after rearranging the numbers in ascending order and descending order.
2. Subtract the smaller number from the larger number
3. Repeat with the obtained number..

For example, when $n = 3421$, we can apply the procedure as follows:

- $4321 - 1234 = 3087$
- $8730 - 0378 = 8352$
- $8532 - 2358 = 6174$

In this case, we applied the procedure three times.

As such, we need to write a function that takes an **integer** and returns an **integer** which represents how many steps we had to take to reach Kaprekar's constant. If the input does not respect rule number 1, then return -1.



Example:

Input: 3421

Output: 3

16. Arrange the pieces - 60 points

Assuming you have an $2 \times N$ board, you need to completely cover the board with either of these two shapes:

- A 2×1 rectangle piece: 
- An L shaped piece: 

For example, if $n = 4$, **ONE** of the ways of filling the board would be like this:



Where the red and green pieces are L shaped and the blue piece is a 2×1 rectangle. If $n \leq 0$, return 0.

Given an integer n , implement a function that returns **ALL** the number of ways we can fill the board..