Problem 1:

Given a string s consisting of words and spaces, return the length of the last word in the string.

A word is a maximal substring consisting of non-space characters only.

Example 1:

Input: s = "Hello World"

Output: 5

Explanation: The last word is "World" with length 5.

Example 2:

Input: s = " fly me to the moon "

Output: 4

Explanation: The last word is "moon" with length 4.

Example 3:

Input: s = "luffy is still joyboy"

Output: 6

Explanation: The last word is "joyboy" with length 6.

Constraints:

1 <= s.length <= 104

s consists of only English letters and spaces ' '.

There will be at least one word in s.

Solution To Problem 1:

Language: javascript

```
let find_last_word_length = (s) => {

    // apply the constraints
    if (s.length < 1) {
        console.log('String cannot be empty');
        return false;
    }else if(s.length > 104){
        console.log('String cannot be over than 104 character');
        return false;
    }
    // find the last word
    let words = s.trim().split(/\s+/);
    let last_word = words[words.length-1];
    console.log(last_word);

    // return it's length
    return last_word.length;
}
```

Problem 2:

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

Symbol Value

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

For example, 2 is written as II in Roman numerals, just two one's added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX.

There are six instances where subtraction is used:

I can be placed before V (5) and X (10) to make 4 and 9.

X can be placed before L (50) and C (100) to make 40 and 90.

C can be placed before D (500) and M (1000) to make 400 and 900.

Given an integer, convert it to a roman numeral.

Example 1:

Input: num = 3

Output: "III"

Explanation: 3 is represented as 3 ones.

Example 2:

Input: num = 58

Output: "LVIII"

Explanation: L = 50, V = 5, III = 3.

Example 3:

Input: num = 1994

Output: "MCMXCIV"

Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.

Constraints:

1 <= num <= 3999

Solution To Problem 2:

Language : javascript

```
function romanOf(value) {
        let numbers = {
            'I': 1,
            'V': 5,
            'X': 10,
            'L': 50,
            'C': 100,
            'D': 500,
            'M': 1000,
        }
        return Object.keys(numbers).find(key => numbers[key] === value);
    }

let convertToRoman = (n) =>{
    n = Number.parseInt(n);

    // constraint
    if(n<0 || n>3999){
        console.log('please insert a number between 1 and 3999');
        return false;
    }

    let roman = '';

    while(n>0){
```

```
if(n>=1000){
        n -= 1000;
        roman += romanOf(1000);
}else if(n>=500){
        if (n>=900) {
                n -= 900;
                roman += 'CM';
        }else{
                n -= 500;
                roman += romanOf(500);
        }
}else if(n>=100){
        if (n>=400) {
                n -= 400;
                roman += 'CD';
        }else{
                n -= 100;
                roman += romanOf(100);
        }
}else if(n>=50){
        if (n>=90) {
                n -= 90;
                roman += 'XC';
        }else{
                n -= 50;
                roman += romanOf(50);
        }
}else if(n>=10){
```

```
                if (n>=40) {
                        n -= 40;
                        roman += 'XL';
                }else{
                        n -= 10;
                        roman += romanOf(10);
                }
        }else if(n>=5){
                if(n == 9){
                        n -= 9;
                        roman += 'IX';
                }else{
                        n -= 5;
                        roman += romanOf(5);
                }
        }else{
                if(n == 4){
                        n -= 4;
                        roman += 'IV';
                }else{
                        n -= 1;
                        roman += 'I';
                }
        }
    }
    return roman;
}
```

Problem 3:

You are choreographing a circus show with various animals. For one act, you are given two

kangaroos on a number line ready to jump in the positive direction (i.e, toward positive infinity).

The first kangaroo starts at location x1 and moves at a rate of v1 meters per jump.

The second kangaroo starts at location x2 and moves at a rate of v2 meters per jump.

You have to figure out a way to get both kangaroos at the same location at the same time as part of

the show. If it is possible, return YES, otherwise return NO.

Example

X1 = 2

V1 = 1

X2 = 1

V2 = 2

After one jump, they are both at x=3 , (x1+v1=2,x2+v2=1+2), so the answer is YES.

Function Description

Complete the function kangaroo in the editor below.

kangaroo has the following parameter(s):

int x1, int v1: starting position and jump distance for kangaroo 1

int x2, int v2: starting position and jump distance for kangaroo 2

Returns YES or NO

Input Format

A single line of four space-separated integers denoting the respective values of x1 , v1 , x2 , and v2.

Constraints

0 <= x1 < x2 <= 10000

1 <= v1 <= 10000

1 <= v2 <= 10000

Sample Input 0: **0 3 4 2**

Sample Output 0: **YES**

Solution To Problem 2:

Language : javascript

```
let willKangarooMeet = (inputs) =>{
        inputs = inputs.trim().split(' ')

        let x1 = Number.parseInt(inputs[0]);
        let v1 = Number.parseInt(inputs[1]);
        let x2 = Number.parseInt(inputs[2]);
        let v2 = Number.parseInt(inputs[3]);

        // validation
        if(inputs.length != 4){
                console.log('invalid input')
                return false;
        }
        if(isNaN(x1) || isNaN(x2) || isNaN(v1) || isNaN(v2)){
                console.log('invalid input')
                return false;
        }

        // constraints
        if(x1<0 || x2<0 || x1>x2 || x1 > 10000 || x2 > 10000){
                console.log('invalid input')
                return false;
        }
        if(v1<=0 || v2<=0 || v1>10000 || v2>10000){
```

```
        console.log('invalid input')
        return false;
}



let hasTheyMeet = false;
let distances = [];
let current_distance;
for (var jump = 1;; jump++) {


        // for every jump count their distance
        current_distance = x2 - x1;
        distances.push(current_distance);


        // if both destination value is same return true
        // else false
        if(current_distance == 0){
                console.log('Yes');
                return true;
        }else if(jump >= 3){

                // if the distances doesnt decrease then stop

                if(distances[1] >= distances[0] && distances[2] >= distances[1]){
                        console.log('No');
```

```
                    return false;
                }
            }

            // if distance decreases but doesnt meet, rather leaves other behind
            if(current_distance < 0){
                console.log('No');
                return false;
            }


            // console.log(current_distance)
            x1 += v1;
            x2 += v2;
        }
}
```