There is a large pile of socks that must be paired by color. Given an array of integers representing the color of each sock, determine how many pairs of socks with matching colors there are.

**Example**

There is one pair of color  and one of color . There are three odd socks left, one of each color. The number of pairs is .

**Function Description**

Complete the *sockMerchant* function in the editor below.

sockMerchant has the following parameter(s):

* *int n:* the number of socks in the pile
* *int ar[n]:* the colors of each sock

**Returns**

* *int:* the number of pairs

**Input Format**

The first line contains an integer , the number of socks represented in .  
The second line contains  space-separated integers, , the colors of the socks in the pile.

**Constraints**

* where

**Sample Input**

STDIN Function

----- --------

9 n = 9

10 20 20 10 10 30 50 10 20 ar = [10, 20, 20, 10, 10, 30, 50, 10, 20]

**Sample Output**

3

**Explanation**

Gráfico, Gráfico de burbujas

Descripción generada automáticamente

'use strict';

const fs = require('fs');

process.stdin.resume();

process.stdin.setEncoding('utf-8');

let inputString = '';

let currentLine = 0;

process.stdin.on('data', function(inputStdin) {

    inputString += inputStdin;

});

process.stdin.on('end', function() {

    inputString = inputString.split('\n');

    main();

});

function readLine() {

    return inputString[currentLine++];

}

/\*

 \* Complete the 'sockMerchant' function below.

 \*

 \* The function is expected to return an INTEGER.

 \* The function accepts following parameters:

 \*  1. INTEGER n

 \*  2. INTEGER\_ARRAY ar

 \*/

function sockMerchant(n, ar) {

    ar.sort((a,b)=>a-b);

    let accum = 0;

    for (let i = 0; i < n ; i++ ){

        if (ar[i] === ar[i+1]) {

            accum++;

            i++;

        }

    }

    return accum;

}

function main() {

    const ws = fs.createWriteStream(process.env.OUTPUT\_PATH);

    const n = parseInt(readLine().trim(), 10);

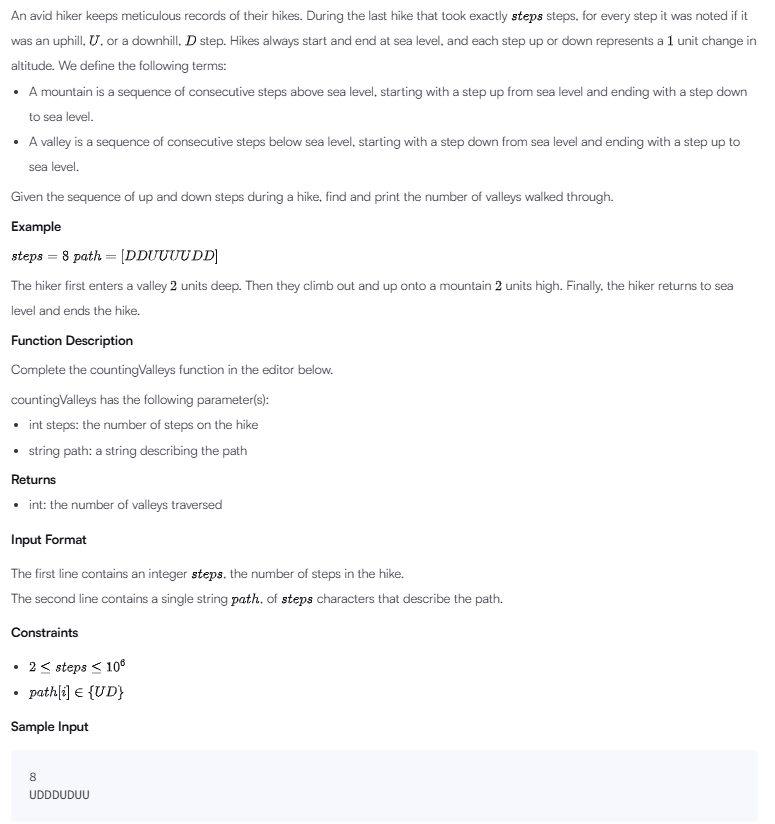
    const ar = readLine().replace(/\s+$/g, '').split(' ').map(arTemp => parseInt(arTemp, 10));

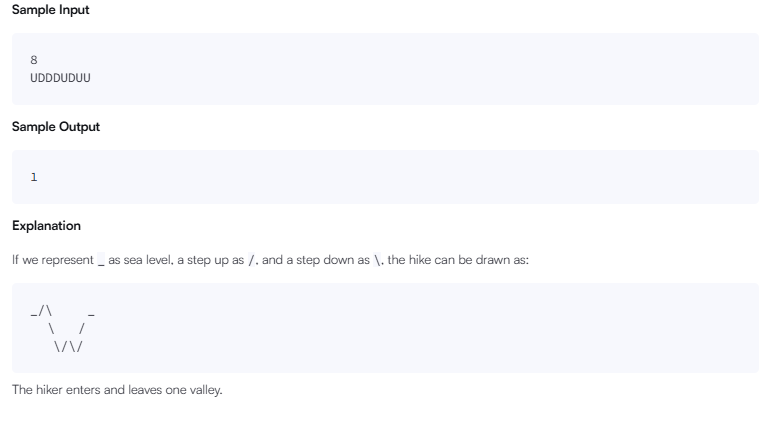
    const result = sockMerchant(n, ar);

    ws.write(result + '\n');

    ws.end();

}





'use strict';

const fs = require('fs');

process.stdin.resume();

process.stdin.setEncoding('utf-8');

let inputString = '';

let currentLine = 0;

process.stdin.on('data', function(inputStdin) {

    inputString += inputStdin;

});

process.stdin.on('end', function() {

    inputString = inputString.split('\n');

    main();

});

function readLine() {

    return inputString[currentLine++];

}

/\*

 \* Complete the 'countingValleys' function below.

 \*

 \* The function is expected to return an INTEGER.

 \* The function accepts following parameters:

 \*  1. INTEGER steps

 \*  2. STRING path

 \*/

function countingValleys(steps, path) {

let seaLvl = 0;

let accValleys = 0;

for (let i = 0; i < steps ; i++ ){

if (path[i] === 'D') {

seaLvl--;

} else {

seaLvl++;

if (seaLvl === 0) {

accValleys++;

}

};

}

return accValleys;

}

function main() {

    const ws = fs.createWriteStream(process.env.OUTPUT\_PATH);

    const steps = parseInt(readLine().trim(), 10);

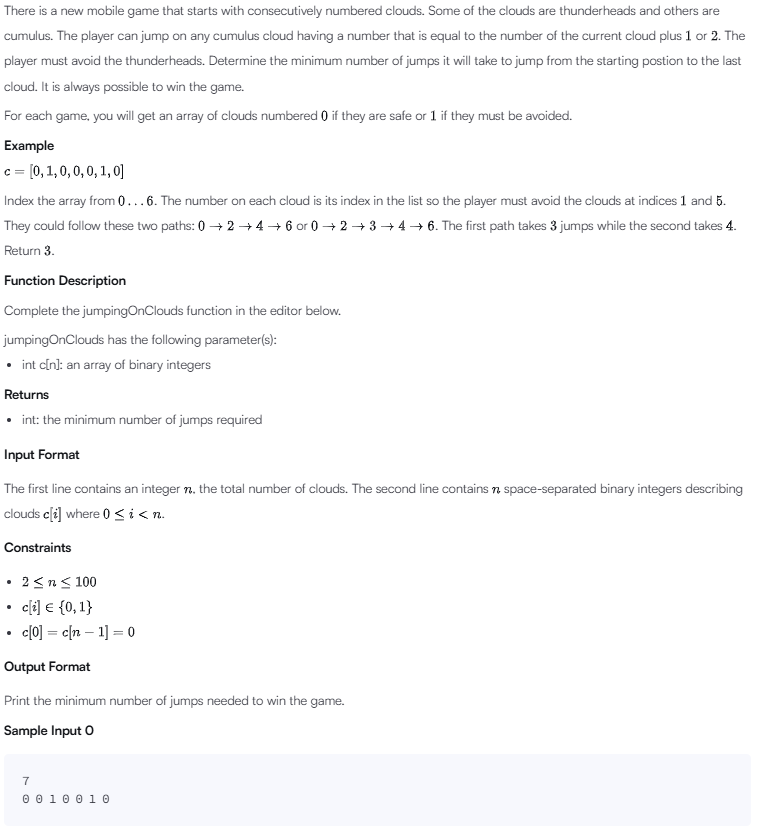
    const path = readLine();

    const result = countingValleys(steps, path);

    ws.write(result + '\n');

    ws.end();

}



Texto

Descripción generada automáticamente con confianza bajaUn dibujo de un perro

Descripción generada automáticamente con confianza mediaAplicación

Descripción generada automáticamente con confianza mediaDiagrama

Descripción generada automáticamente

'use strict';

const fs = require('fs');

process.stdin.resume();

process.stdin.setEncoding('utf-8');

let inputString = '';

let currentLine = 0;

process.stdin.on('data', function(inputStdin) {

    inputString += inputStdin;

});

process.stdin.on('end', function() {

    inputString = inputString.split('\n');

    main();

});

function readLine() {

    return inputString[currentLine++];

}

/\*

 \* Complete the 'jumpingOnClouds' function below.

 \*

 \* The function is expected to return an INTEGER.

 \* The function accepts INTEGER\_ARRAY c as parameter.

 \*/

function jumpingOnClouds(c) {

    let jumps = 0;

    let i;

    for (i = 0; i < c.length; i++) {

            if (c[i+2] === 0){

                i++;

                jumps++;

            } else if (c[i+1] === 0){

                jumps++;

            }

    }

    return jumps;

}

function main() {

    const ws = fs.createWriteStream(process.env.OUTPUT\_PATH);

    const n = parseInt(readLine().trim(), 10);

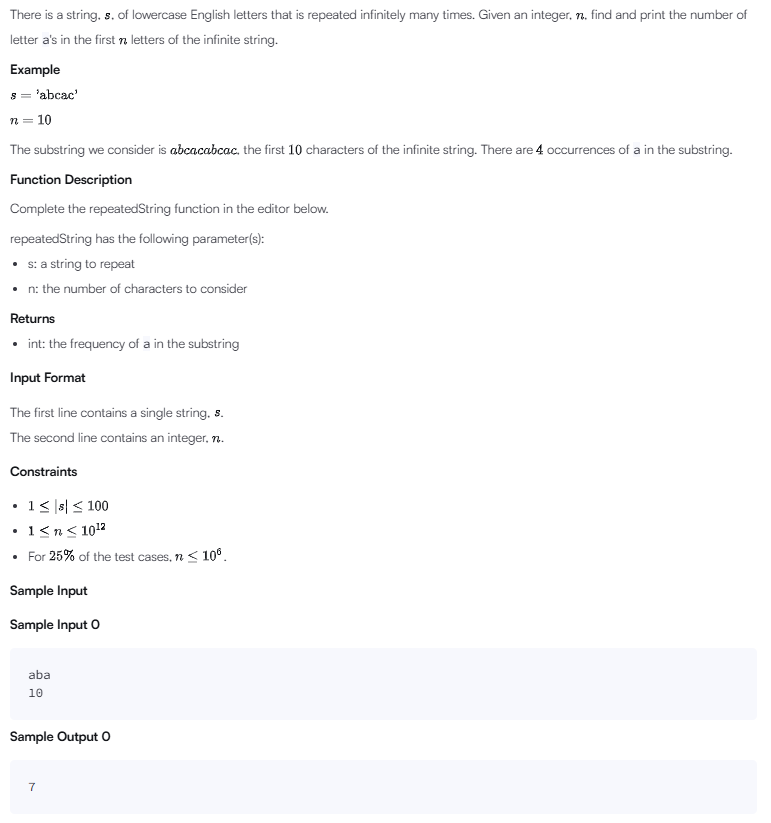
    const c = readLine().replace(/\s+$/g, '').split(' ').map(cTemp => parseInt(cTemp, 10));

    const result = jumpingOnClouds(c);

    ws.write(result + '\n');

    ws.end();

}



Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

'use strict';

const fs = require('fs');

process.stdin.resume();

process.stdin.setEncoding('utf-8');

let inputString = '';

let currentLine = 0;

process.stdin.on('data', function(inputStdin) {

    inputString += inputStdin;

});

process.stdin.on('end', function() {

    inputString = inputString.split('\n');

    main();

});

function readLine() {

    return inputString[currentLine++];

}

/\*

 \* Complete the 'repeatedString' function below.

 \*

 \* The function is expected to return a LONG\_INTEGER.

 \* The function accepts following parameters:

 \*  1. STRING s

 \*  2. LONG\_INTEGER n

 \*/

function repeatedString(s, n) {

    let mod = n%s.length;

    let result;

    let count = 0;

    let countTruck = 0;

    if (mod > 0) {

        for (let k = 0; k < s.length-mod; k++){

            if (s[k] === 'a'){

                countTruck++;

            }

        }

    }

    let i = 0;

    while ( i < s.length ) {

        if (s[i] === 'a') {

          count++;

        };

        i++;

    }

    result = count \* Math.floor(n/s.length)+countTruck;

    return result;

}

function main() {

    const ws = fs.createWriteStream(process.env.OUTPUT\_PATH);

    const s = readLine();

    const n = parseInt(readLine().trim(), 10);

    const result = repeatedString(s, n);

    ws.write(result + '\n');

    ws.end();

}