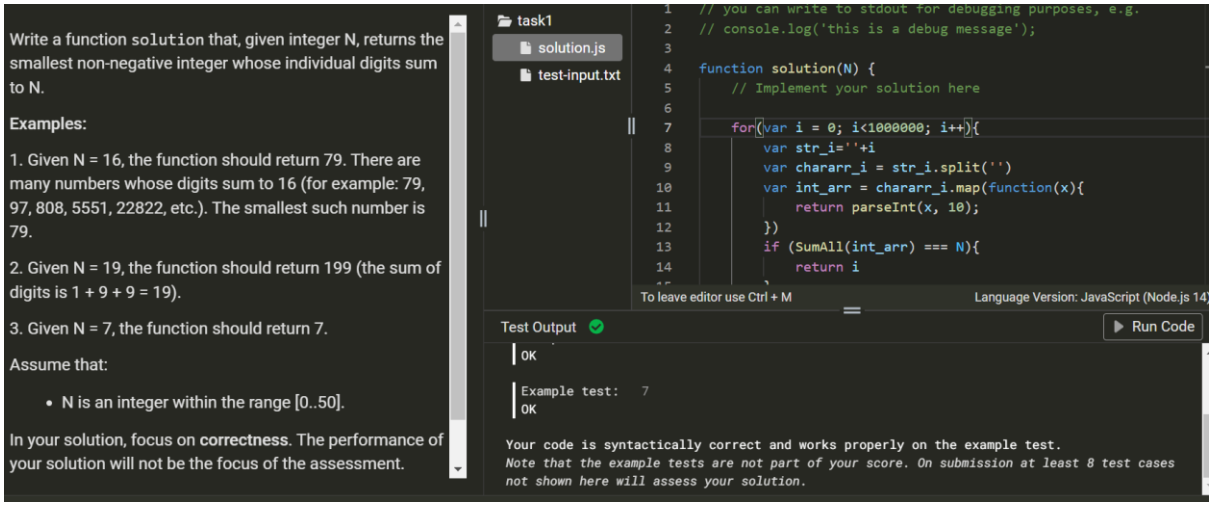


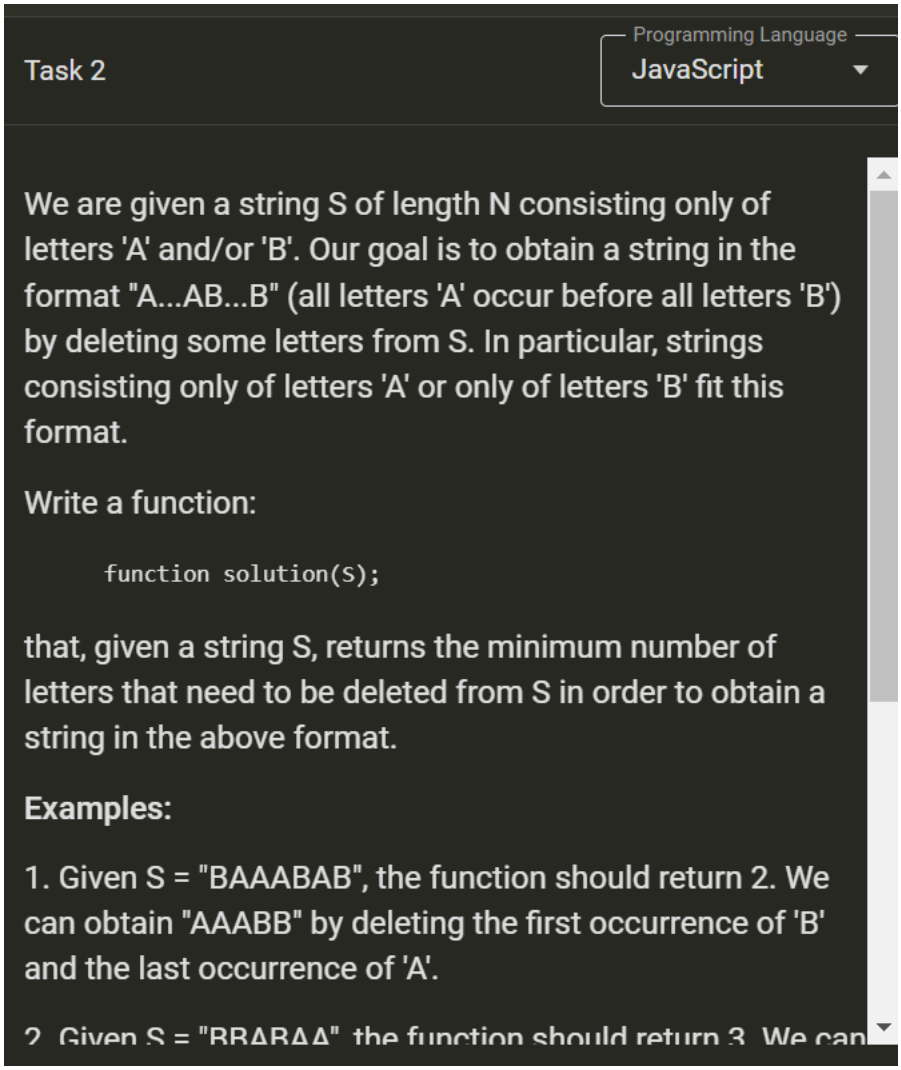

```javascript
// you can write to stdout for debugging purposes, e.g.
// console.log('this is a debug message');

function solution(N) {
    // Implement your solution here

    for(var i = 0; i<1000000; i++){
        var str_i=''+i
        var chararr_i = str_i.split('')
        var int_arr = chararr_i.map(function(x){
            return parseInt(x, 10);
        })
        if (SumAll(int_arr) === N){
            return i
        }
    }
}

function SumAll(A){
    var result = 0
    for(var i = 0; i < A.length; i++){
        result += A[i]
    }
    return result
}
```

Note: Incorrect solution



Examples:

1. Given S = "BAAABAB", the function should return 2. We can obtain "AAABB" by deleting the first occurrence of 'B' and the last occurrence of 'A'.

2. Given S = "BBABAA", the function should return 3. We can delete all occurrences of 'A' or all occurrences of 'B'.

3. Given S = "AABBBB", the function should return 0. We do not have to delete any letters, because the given string is already in the expected format.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..100,000];
- string S is made only of the characters 'A' and/or 'B'.


```
// you can write to stdout for debugging purposes, e.g.
// console.log('this is a debug message');

function solution(S) {
    S = S.split('');
    var sA  = true;
    var action = 0;
    var A=[]
    var B=[]
    var B_l
    for(var i=0; i< S.length; i++){
        if(S[i]==='A'){
            A.push(i)
        } else {
            B.push(i)
        }
    }

    var check_A = S[S.length/2-1] === 'A'
    var check_B = S[S.length/2] === 'B'
    B_l=B.length

    //This while block is just a cope
    while(A[A.length-1]>B[0]){

        if(B[0]<A[0]){
            B.shift()
```

```
            action++
            continue
        }/* else if(A[A.length-1]>B[B.length-1]){
            A.pop()
        }*/
        else{
            A.pop()
        }
        action++
    }
    //nothing about this conditional makes any sense.
    if(B.length === A.length && check_A && check_B){
        return B_l
    }
    return action
}
```

Task 3

Programming Language: JavaScript

A garden is divided into N sections numbered from 0 to N-1. It is described by an array A, where A[K] denotes the number of trees in the K-th section. To make the garden look more organized, we want the number of trees in every section to be the same. As we don't want to cut any trees down, we can perform either of the following actions:

- planting a new tree in one of the sections;
- replanting an existing tree, moving it from one section to another.

We want to minimize the number of actions performed.

Write a function:

```
function solution(A);
```

that, given an array A consisting of N integers describing the garden, returns the minimum number of actions we need to perform in order to make all sections of the garden contain the same number of trees.

that, given an array A consisting of N integers describing the garden, returns the minimum number of actions we need to perform in order to make all sections of the garden contain the same number of trees.

Examples:

1. Given A = [1, 2, 2, 4] the function should return 4. We can move one tree from A[3] to A[1] and obtain A = [1, 3, 2, 3]. Then we can plant two trees in A[0] and one tree in A[2] to make every section contain three trees.

2. Given A = [4, 2, 4, 6], the function should return 2. We can move two trees from A[3] to A[1]. This way, every section in the garden will contain four trees.

3. Given A = [1, 1, 2, 1], the function should return 3. We can plant one tree in A[0], A[1] and A[3] so that each section in the garden contains two trees.

Write an **efficient** algorithm for the following assumptions:

3. Given A = [1, 1, 2, 1], the function should return 3. We can plant one tree in A[0], A[1] and A[3] so that each section in the garden contains two trees.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..100,000];
- each element of array A is an integer within the range [1..1,000,000,000];
- the answer is always less than or equal to 2,000,000,000.


```
// you can write to stdout for debugging purposes, e.g.
// console.log('this is a debug message');

function solution(A) {
    // Implement your solution here
    A = A.sort(function(a,b){return b-a})
    var reserveTrees = 0
```

```
    var actions = 0
    var AllTrees = 0
    for(var i = 0; i<A.length ; i++){
        AllTrees += A[i]
    }
    while(AllTrees%A.length !== 0){
        AllTrees++
    }
    var treesPerSection = AllTrees/A.length
    //console.log("trees per section:" + treesPerSection)
    for(var i = 0; i<A.length ; i++){
        //console.log(A[i])
        if (A[i]>treesPerSection){
            reserveTrees+=A[i]-treesPerSection
            A[i]-=A[i]-treesPerSection
        }
        //console.log("reserveTrees:" + reserveTrees)
        if(A[i]<treesPerSection){
            actions+=treesPerSection-A[i]
            A[i]++
        }
        /*while(A[i]<treesPerSection){
            if(reserveTrees>0 && A[i]<treesPerSection){
                reserveTrees--
                actions++
                A[i]++
            } else if(reserveTrees !== 0 ){
                actions++
                A[i]++
            }
        }*/
        //console.log("A[" + i + "]: " + A[i])
    }
    return actions
}
```