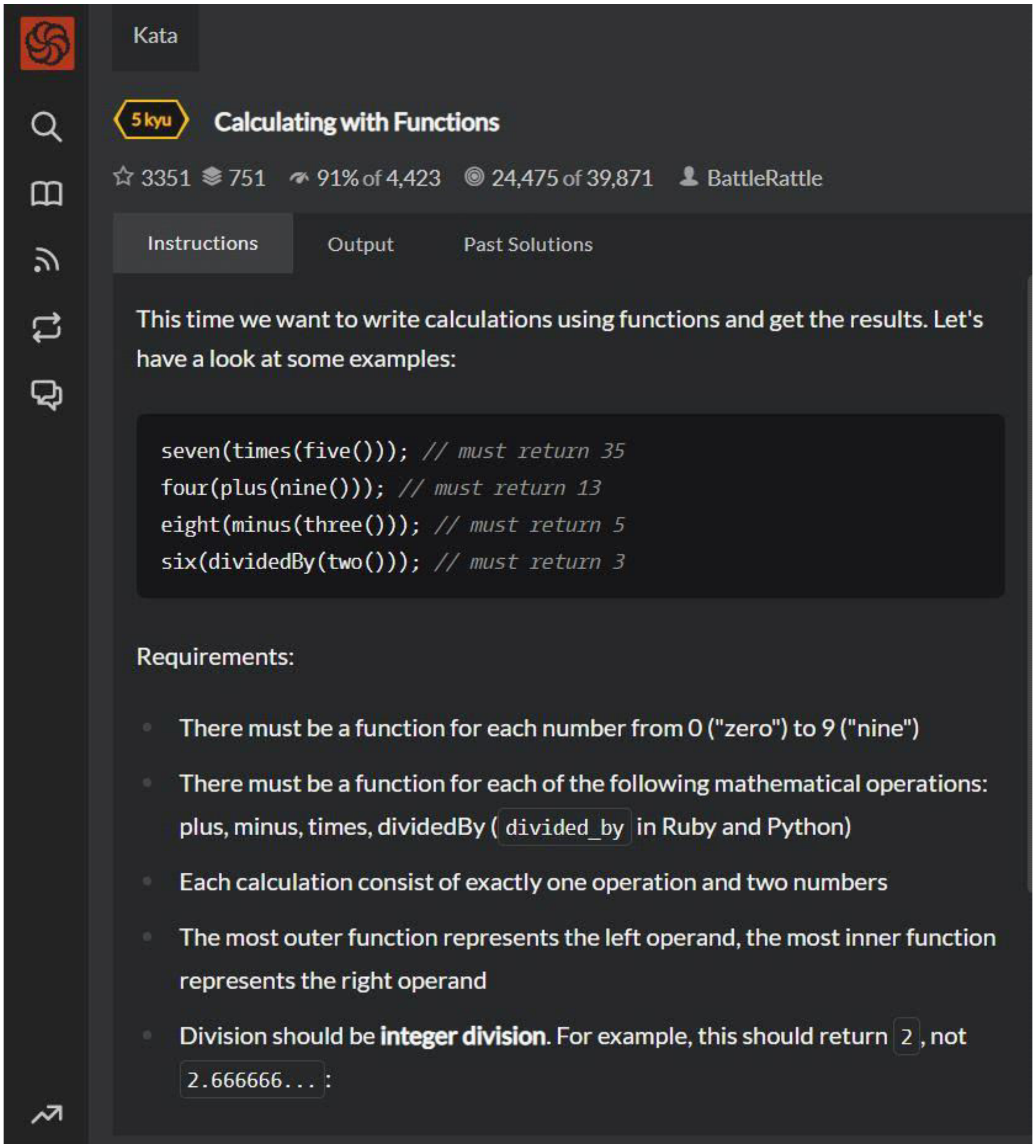
HomeTask :

2.2



5 kyu **Calculating with Functions**

☆ 3351 751 91% of 4,423 24,475 of 39,871 BattleRattle

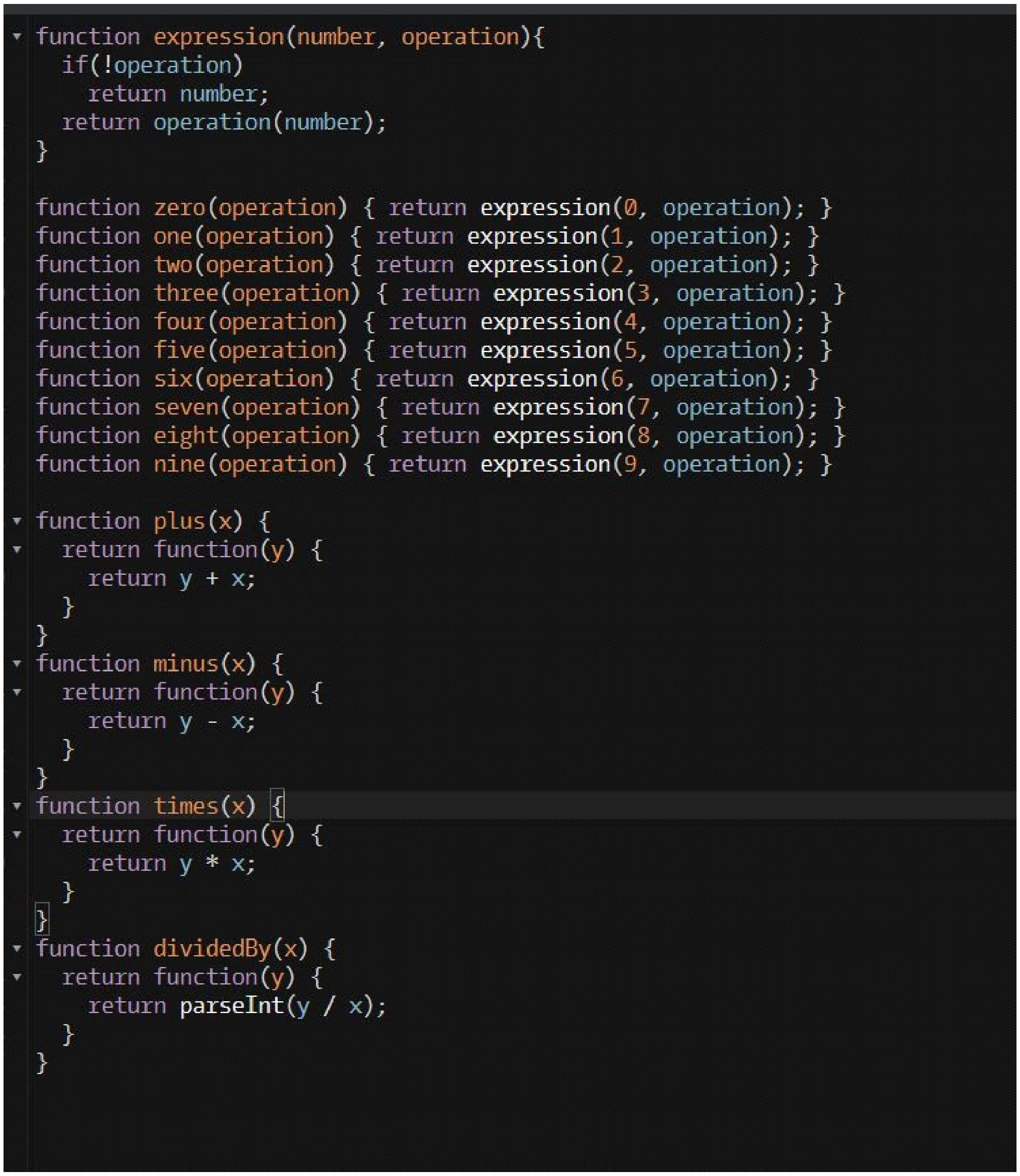
Instructions Output Past Solutions

This time we want to write calculations using functions and get the results. Let's have a look at some examples:

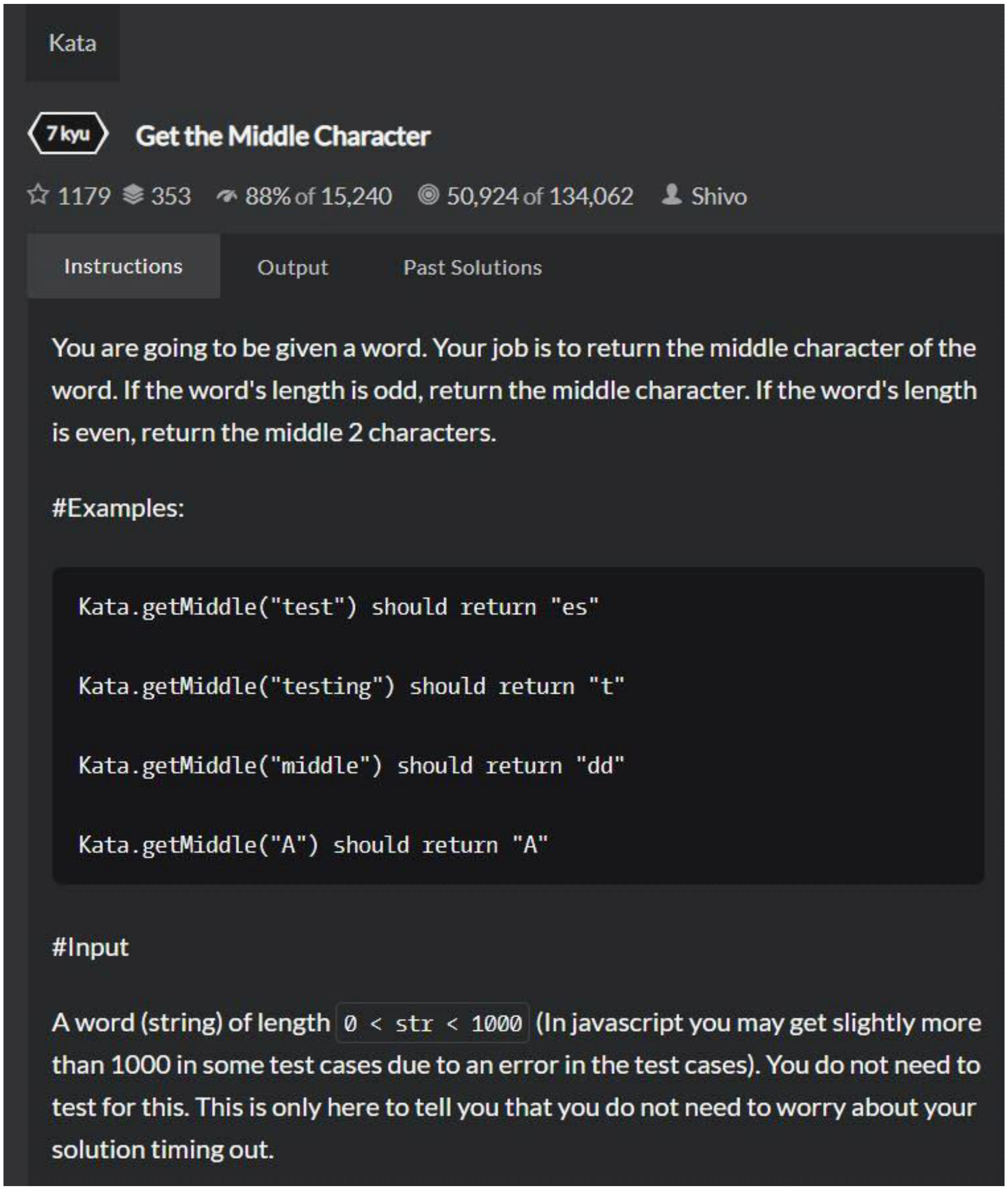
```
seven(times(five())); // must return 35
four(plus(nine())); // must return 13
eight(minus(three())); // must return 5
six(dividedBy(two())); // must return 3
```

Requirements:

- There must be a function for each number from 0 ("zero") to 9 ("nine")
- There must be a function for each of the following mathematical operations: plus, minus, times, dividedBy (`divided_by` in Ruby and Python)
- Each calculation consist of exactly one operation and two numbers
- The most outer function represents the left operand, the most inner function represents the right operand
- Division should be **integer division**. For example, this should return `2`, not `2.666666...`:



```javascript
function expression(number, operation){
  if(!operation)
    return number;
  return operation(number);
}

function zero(operation) { return expression(0, operation); }
function one(operation) { return expression(1, operation); }
function two(operation) { return expression(2, operation); }
function three(operation) { return expression(3, operation); }
function four(operation) { return expression(4, operation); }
function five(operation) { return expression(5, operation); }
function six(operation) { return expression(6, operation); }
function seven(operation) { return expression(7, operation); }
function eight(operation) { return expression(8, operation); }
function nine(operation) { return expression(9, operation); }

function plus(x) {
  return function(y) {
    return y + x;
  }
}
function minus(x) {
  return function(y) {
    return y - x;
  }
}
function times(x) {
  return function(y) {
    return y * x;
  }
}
function dividedBy(x) {
  return function(y) {
    return parseInt(y / x);
  }
}
```


Kata

7 kyu **Get the Middle Character**

☆ 1179 353 88% of 15,240 50,924 of 134,062 Shivo

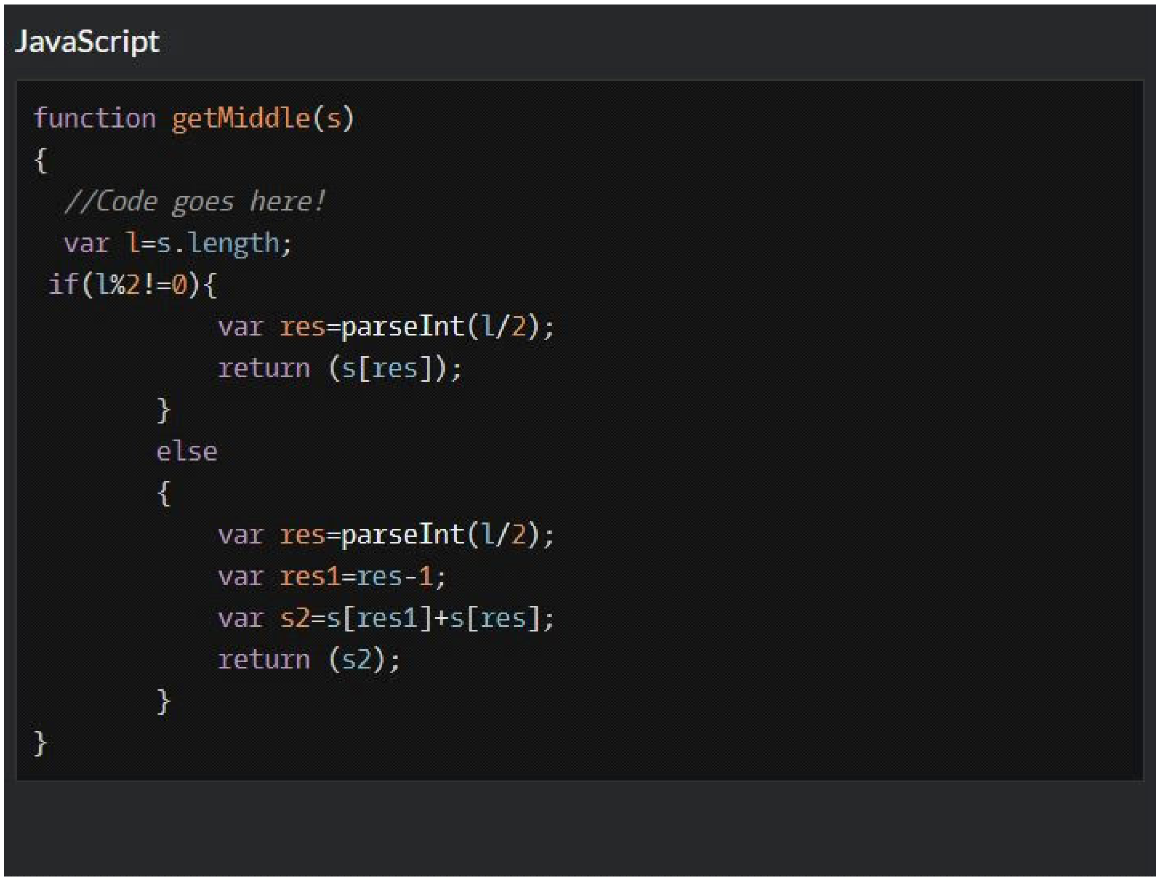
You are going to be given a word. Your job is to return the middle character of the word. If the word's length is odd, return the middle character. If the word's length is even, return the middle 2 characters.

#Examples:

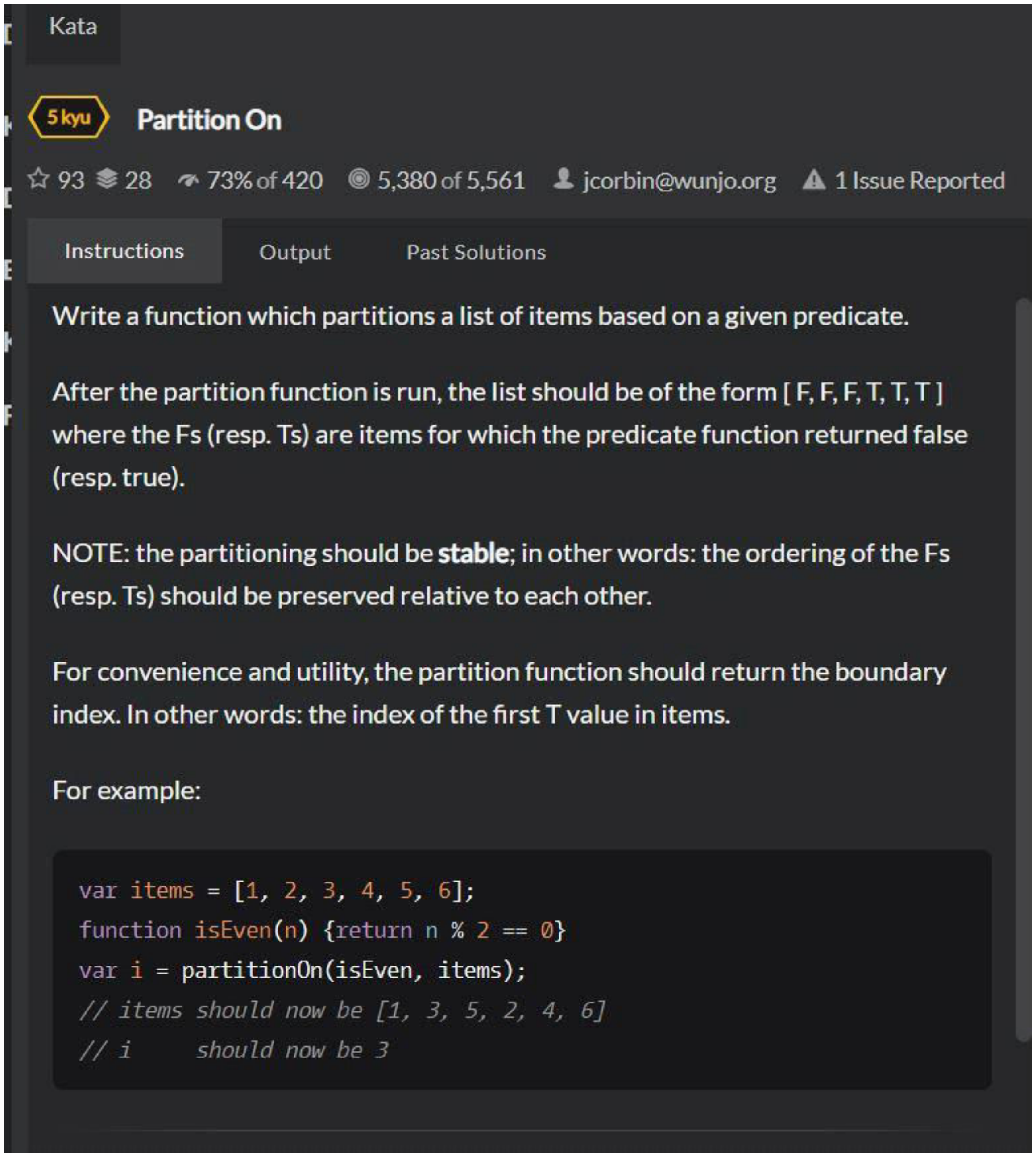
```
Kata.getMiddle("test") should return "es"

Kata.getMiddle("testing") should return "t"

Kata.getMiddle("middle") should return "dd"

Kata.getMiddle("A") should return "A"
```

#Input

A word (string) of length `0 < str < 1000` (In javascript you may get slightly more than 1000 in some test cases due to an error in the test cases). You do not need to test for this. This is only here to tell you that you do not need to worry about your solution timing out.


JavaScript

```
function getMiddle(s)
{
  //Code goes here!
  var l=s.length;
 if(l%2!=0){
            var res=parseInt(l/2);
            return (s[res]);
        }
        else
        {
            var res=parseInt(l/2);
            var res1=res-1;
            var s2=s[res1]+s[res];
            return (s2);
        }
}
```


Kata
5 kyu
☆ 93 28 73% of 420 5,380 of 5,561 jcorbin@wunjo.org 1 Issue Reported
Instructions
Output
Past Solutions

Partition On

Write a function which partitions a list of items based on a given predicate.

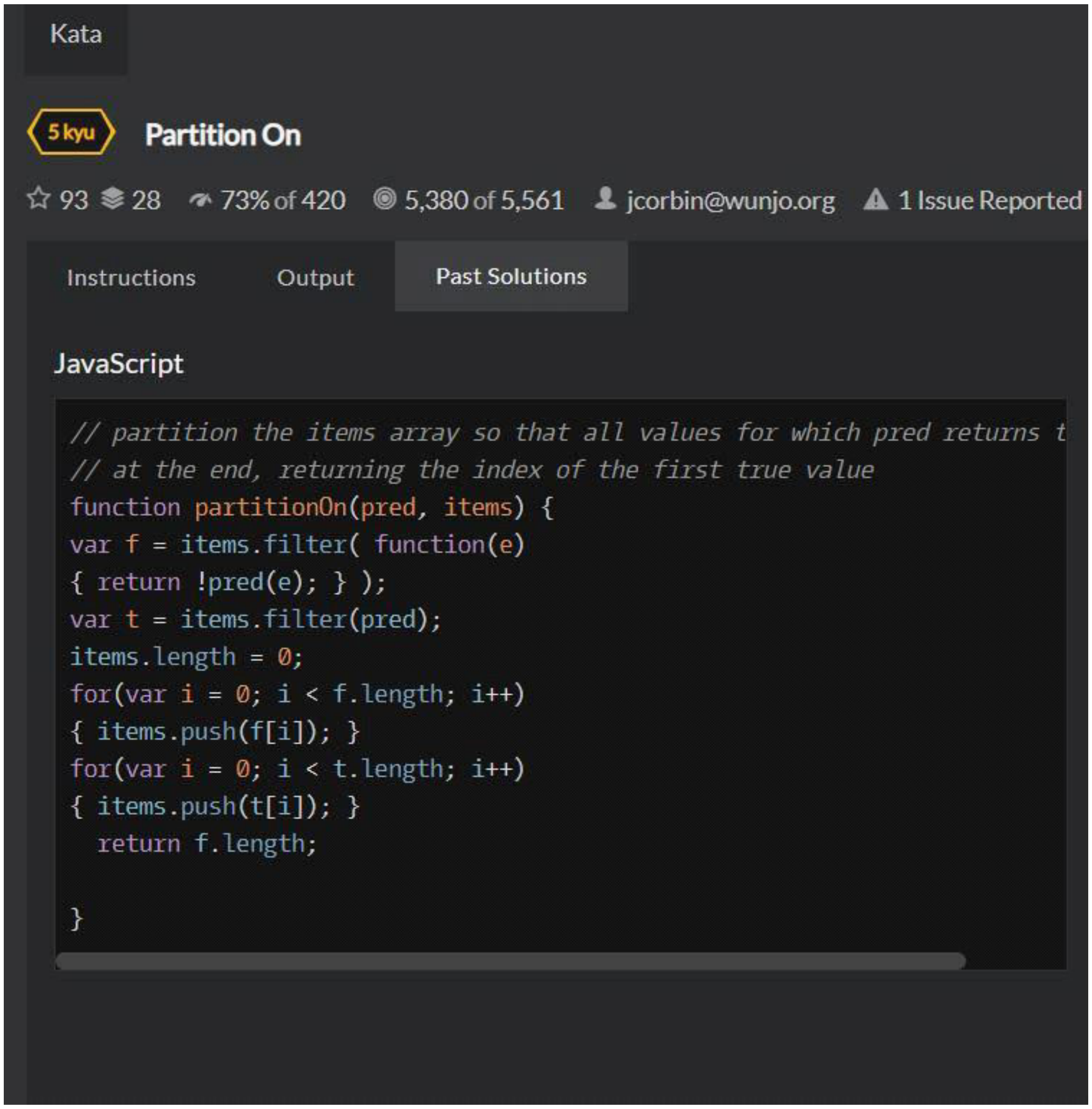
After the partition function is run, the list should be of the form [F, F, F, T, T, T] where the Fs (resp. Ts) are items for which the predicate function returned false (resp. true).

NOTE: the partitioning should be **stable**; in other words: the ordering of the Fs (resp. Ts) should be preserved relative to each other.

For convenience and utility, the partition function should return the boundary index. In other words: the index of the first T value in items.

For example:

```
var items = [1, 2, 3, 4, 5, 6];
function isEven(n) {return n % 2 == 0}
var i = partitionOn(isEven, items);
// items should now be [1, 3, 5, 2, 4, 6]
// i     should now be 3
```

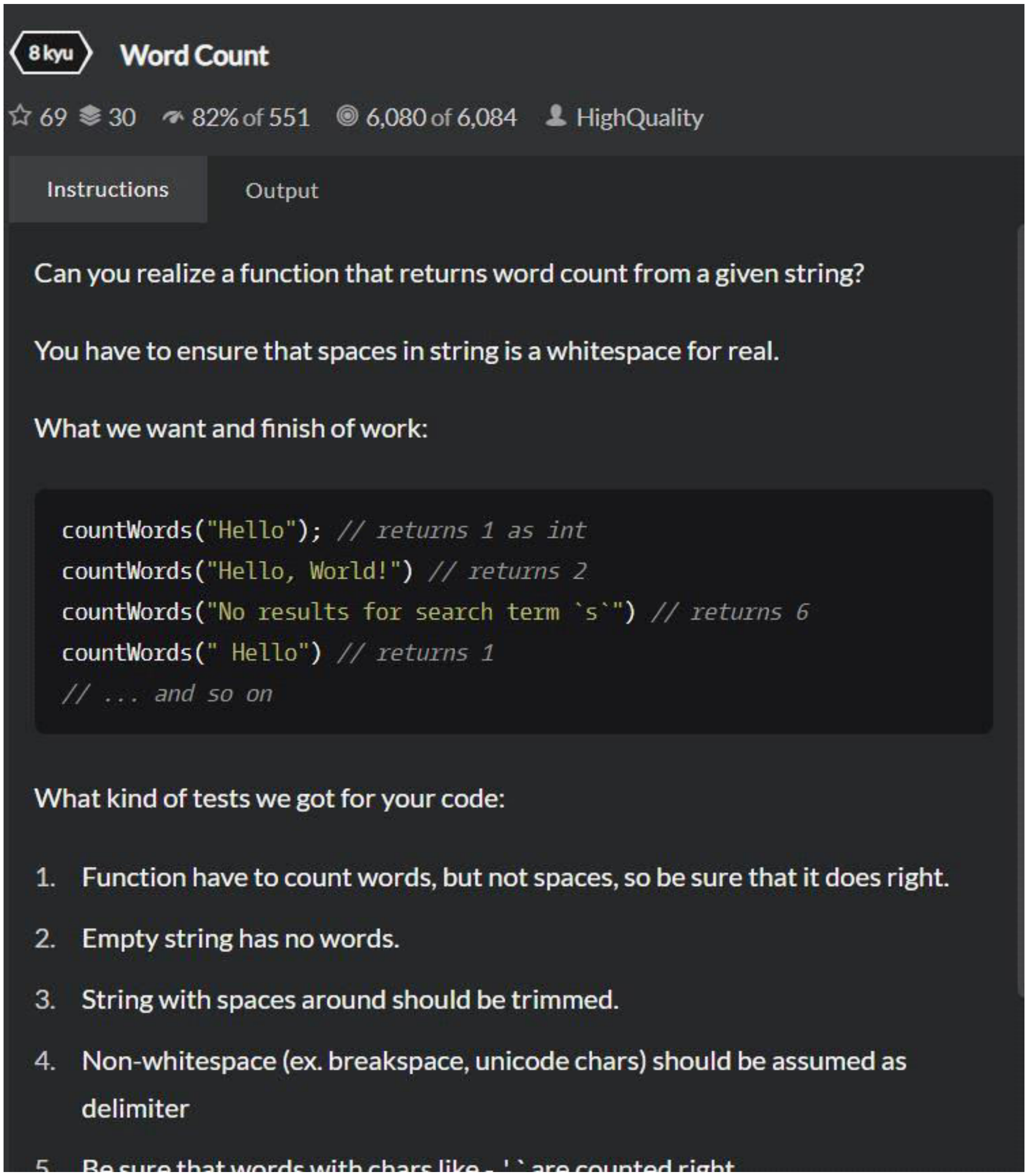


5 kyu **Partition On**

☆ 93 28 73% of 420 5,380 of 5,561 jcorbin@wunjo.org 1 Issue Reported

JavaScript

```
// partition the items array so that all values for which pred returns t
// at the end, returning the index of the first true value
function partitionOn(pred, items) {
var f = items.filter( function(e)
{ return !pred(e); } );
var t = items.filter(pred);
items.length = 0;
for(var i = 0; i < f.length; i++)
{ items.push(f[i]); }
for(var i = 0; i < t.length; i++)
{ items.push(t[i]); }
  return f.length;

}
```


8 kyu **Word Count**

☆ 69 30 82% of 551 6,080 of 6,084 HighQuality

Instructions Output

Can you realize a function that returns word count from a given string?

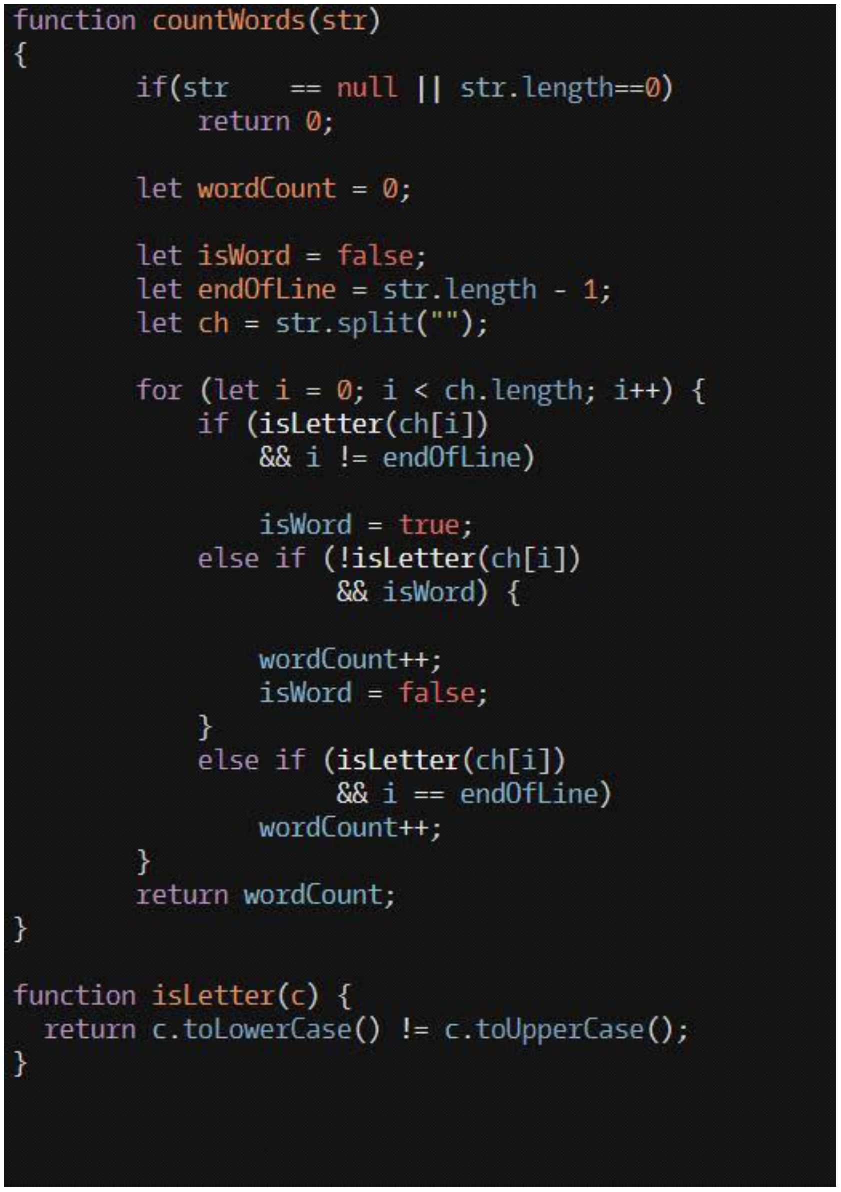
You have to ensure that spaces in string is a whitespace for real.

What we want and finish of work:

```
countWords("Hello"); // returns 1 as int
countWords("Hello, World!") // returns 2
countWords("No results for search term `s`") // returns 6
countWords(" Hello") // returns 1
// ... and so on
```

What kind of tests we got for your code:

1. Function have to count words, but not spaces, so be sure that it does right.
2. Empty string has no words.
3. String with spaces around should be trimmed.
4. Non-whitespace (ex. breakspace, unicode chars) should be assumed as delimiter
5. Be sure that words with chars like - ' ` are counted right



```
function countWords(str)
{
        if(str    == null || str.length==0)
            return 0;

        let wordCount = 0;

        let isWord = false;
        let endOfLine = str.length - 1;
        let ch = str.split("");

        for (let i = 0; i < ch.length; i++) {
            if (isLetter(ch[i])
                && i != endOfLine)

                isWord = true;
            else if (!isLetter(ch[i])
                     && isWord) {

                wordCount++;
                isWord = false;
            }
            else if (isLetter(ch[i])
                     && i == endOfLine)
                wordCount++;
        }
        return wordCount;
}

function isLetter(c) {
  return c.toLowerCase() != c.toUpperCase();
}
```