

First task

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Finish the solution so that it returns the sum of all the multiples of 3 or 5 below the number passed in.

Additionally, if the number is negative, return 0.

Note: If the number is a multiple of both 3 and 5, only count it once.

```
function solution(n){
  let sum = 0;
  for (let i = 0; i < n; i++) {
    if (i % 5 == 0 || i % 3 == 0) {
      sum = sum + i;
    }
  }
  return sum;
}
```

The screenshot shows a coding challenge interface for the problem "Multiples of 3 or 5". The interface includes a header with the problem name, a star rating (3504), a difficulty level (6kyu), and a progress indicator (88% of 36,321). The main content area is divided into two panels: "Instructions" and "Output". The "Instructions" panel displays the problem description and the provided code snippet. The "Output" panel shows the test results, including a list of test cases (basic tests, smallest cases, random cases) and their corresponding results (passed/failed). The "Solution" panel displays the user's code, which is a JavaScript function that calculates the sum of multiples of 3 or 5 below a given number. The "Sample Tests" panel shows the test cases used to verify the solution. The interface also includes a "SKIP" button, a "UNLOCK SOLUTIONS" button, a "DISCUSS (515)" button, a "RESET" button, a "TEST" button, and a "SUBMIT" button.

6kyu Multiples of 3 or 5

☆ 3504 638 88% of 36,321 114,555 of 366,628 jhoffner
▲ 2 Issues Reported

Instructions Output

Time: 1080ms Passed: 111 Failed: 0

Test Results:

- basic tests
 - n=10
 - n=20
 - n=200Completed in 1ms
- smallest cases
 - n=-1
 - n=0
 - n=1
 - n=2
 - n=3
 - n=4
 - n=5
 - n=6Completed in 2ms
- random cases
 - n=-5
 - n=-9
 - n=597
 - n=5199
 - n=397
 - n=96593

Solution

```
1 function solution(n){
2   let sum = 0;
3   for (let i = 0; i < n; i++) {
4     if (i % 5 == 0 || i % 3 == 0) {
5       sum = sum + i;
6     }
7   }
8   return sum;
9 }
```

Excellent! You may take your time to refactor/comment your solution. Submit when ready.

Sample Tests

```
1 const { assert } = require("chai")
2
3 function test(n, expected) {
4   it(`n=${n}`, () => {
5     let actual = solution(n)
6     assert.strictEqual(actual, expected)
7   })
8 }
9
10 describe("basic tests", function(){
11   test(10,23)
12 })
```

SKIP UNLOCK SOLUTIONS DISCUSS (515) RESET TEST SUBMIT

Second task

Your task, is to calculate the minimal number of moves to win the game "Towers of Hanoi", with given number of disks.

Towers of Hanoi is a simple game consisting of three rods, and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape.

The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

Only one disk can be moved at a time.

Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.

No disk may be placed on top of a smaller disk.

```
var hanoi = function(n) {  
  const steps = Array(n + 1).fill(0);  
  steps[0] = 0;  
  steps[1] = 1;  
  
  for (let i = 2; i <= n; i++) {  
    steps[i] = steps[i - 1] * 2 + 1;  
  }  
  
  return steps[n];  
};  
  
};
```

6 kyu

Hanoi record ✓

☆ 95 🏆 30 🔄 85% of 370 🌐 939 of 2,740 👤 Bugari

InstructionsOutputPast Solutions

Time: 860ms Passed: 2 Failed: 0

Test Results:
Basic tests
Testing for fixed tests
Random tests
Completed in 2ms

You have passed all of the tests! :)

JavaScriptNode v18.xVIMEMACS

Solution

```
1 var hanoi = function(n) {  
2   const steps = Array(n + 1).fill(0);  
3   steps[0] = 0;  
4   steps[1] = 1;  
5  
6   for (let i = 2; i <= n; i++) {  
7     steps[i] = steps[i - 1] * 2 + 1;  
8   }  
9  
10  return steps[n];  
11 };
```

Good Job! You may take your time to refactor/comment your solution. Submit when ready.

Sample Tests

```
1 const chai = require("chai");  
2 const assert = chai.assert;  
3 chai.config.truncateThreshold=0;  
4  
5 describe("Basic tests", () => {  
6   it("Testing for fixed tests", () => {  
7     assert.strictEqual(hanoi(1), 1)  
8     assert.strictEqual(hanoi(2), 3)  
9     assert.strictEqual(hanoi(3), 7)  
10    assert.strictEqual(hanoi(4), 15)  
11  })  
12 })
```

SKIPVIEW SOLUTIONSDISCUSS (30)RESETTESTSUBMIT

Third task

Your task is to construct a building which will be a pile of n cubes. The cube at the bottom will have a volume of n^3 , the cube above will have volume of $(n-1)^3$ and so on until the top which will have a volume of 1^3 .

You are given the total volume m of the building. Being given m can you find the number n of cubes you will have to build?

The parameter of the function `findNb` (`find_nb`, `find-nb`, `findNb`, ...) will be an integer m and you have to return the integer n such as $n^3 + (n-1)^3 + (n-2)^3 + \dots + 1^3 = m$ if such a n exists or `-1` if there is no such n .

Examples:

`findNb(1071225) --> 45`

`findNb(91716553919377) --> -1`

```
function findNb(m) {
  let n = 0;
  if (m < 0) {
    console.log('The variable "m" must be positive')
  }
  while (n < m) {
    n++;
    m = m - n ** 3;
  }
  if (m == 0) {
    return n;
  }
  else {
    return -1;
  }
}
```

6 keys

Build a pile of Cubes

☆ 2715 406 86% of 7,867 21,100 of 77,919 g964

JavaScript

Node v18.x

VIM EMACS

Instructions

Output

Time: 876ms Passed: 101 Failed: 0

Test Results:

Basic tests

Random tests

Testing for : 16320816249409

Testing for : 833

Testing for : 5336908530625

Testing for : 942

Testing for : 315630757

Testing for : 764

Testing for : 114972533777

Testing for : 918

Testing for : 189031909284

Testing for : 351

Testing for : 193786612367076

Testing for : 736

Testing for : 16902606793284

Testing for : 264

Testing for : 111658227410

Testing for : 300

Solution

```
1 function findNb(m) {
2   let n = 0;
3   if (m < 0) {
4     console.log('The variable "m" must be positive')
5   }
6   while (n < m) {
7     n++;
8     m = m - n ** 3;
9   }
10  if (m == 0) {
11    return n;
12  }
13  else {
14    return -1;
15  }
16 }
17
```

Correctamundo! You may take your time to refactor/comment your solution. Submit when ready.

Sample Tests

```
1 const { assert } = require('chai');
2
3 it('Basic tests', function() {
4   assert.strictEqual(findNb(4183059834009), 2022)
5   assert.strictEqual(findNb(24723578342962), -1)
6   assert.strictEqual(findNb(135440716410000), 4824)
7   assert.strictEqual(findNb(40539911473216), 3568)
8 })
9
10
11
```

Fourth task

Define a function that takes an integer argument and returns a logical value true or false depending on if the integer is a prime.

Per Wikipedia, a prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself.

Requirements:

- You can assume you will be given an integer input.

- You can not assume that the integer will be only positive. You may be given negative numbers as well (or 0).

- NOTE on performance: There are no fancy optimizations required, but still the most trivial solutions might time out. Numbers go up to 2^{31} (or similar, depending on language). Looping all the way up to n , or $n/2$, will be too slow.

```
function isPrime(number) {
  if (number <= 1) {
    return false;
  }
  if (number <= 3) {
    return true;
  }
  if (number % 2 === 0 || number % 3 === 0) {
    return false;
  }

  for (let i = 5; i * i <= number; i += 6) {
    if (number % i === 0 || number % (i + 2) === 0) {
      return false;
    }
  }

  return true;
}
```

6 kyu

Is a number prime?

☆ 2240 🏆 407 📈 84% of 8,214 📊 38,422 of 99,440 🧑 arithmetric 🚩 2 Issues Reported

Instructions Output

Time: 1069ms Passed: 4 Failed: 0

Test Results:

Basic

- Basic tests
- Test prime
- Test not prime

Completed in 1ms

Random tests

- Random tests

Completed in 69ms

You have passed all of the tests! :)

JavaScript Node v18.x VIM EMACS

Solution

```
1
2 function isPrime(number) {
3   if (number <= 1) {
4     return false;
5   }
6   if (number <= 3) {
7     return true;
8   }
9   if (number % 2 === 0 || number % 3 === 0) {
10    return false;
11  }
12
13  for (let i = 5; i * i <= number; i += 6) {
14    if (number % i === 0 || number % (i + 2) === 0) {
15      return false;
16    }
17  }
18
19  return true;
20 }
```

Sample Tests

```
3 describe("Basic", () => {
4
5   it("Basic tests", () => {
6
7     Test.assertEquals(isPrime(0), false, "0 is not prime");
8     Test.assertEquals(isPrime(1), false, "1 is not prime");
9     Test.assertEquals(isPrime(2), true, "2 is prime");
10    Test.assertEquals(isPrime(73), true, "73 is prime");
11    Test.assertEquals(isPrime(75), false, "75 is not prime");
12    Test.assertEquals(isPrime(-1), false, "-1 is not prime");
13  })
14 }
```

Fifth task

In this little assignment you are given a string of space separated numbers, and have to return the highest and lowest number.

Examples:

```
highAndLow("1 2 3 4 5"); // return "5 1"
```

```
highAndLow("1 2 -3 4 5"); // return "5 -3"
```

```
highAndLow("1 9 3 4 -5"); // return "9 -5"
```

```
function highAndLow(str){
  let max = "";
  let min = "";
  let num = "";
  for (let i = 0; i < str.length; i++) {
    let digit = str[i];

    if (digit !== ' ') {
      if (str[i + 1] === ' ' || str[i + 1] === undefined) {
        num = num + digit;
        if (max === " " || min === " ") {
          max = num;
          min = num;
        }
        if (Number(num) >= max) {
          max = Number(num);
        }
        if (Number(num) <= min) {
          min = Number(num);
        }
        num = "";
      } else if (str[i + 1] !== ' ') {
        num = num + digit;
      }
    }
  }
  return `${max} ${min}`
}
```

7knu

Highest and Lowest

☆ 2971

👤 664

📊 90% of 21,774

👤 105,550 of 256,487

👤 Deantwo

Instructions

Output

Time: 814ms

Passed: 10

Failed: 0

Test Results:

Fixed tests

Testing basic input

Testing common string sort mistake

Testing positive and negative

Testing positive and positive

Testing negative and negative

Testing positive, negative and zero

Testing positive, positive and zero

Testing negative, negative and zero

Testing single

Completed in 3ms

Random tests

Testing for 100 random tests

Completed in 5ms

You have passed all of the tests! :)

JavaScript

Node v18.x

VIM

EMACS

🔍

Solution

```
1 function highAndLow(str){
2   let max = '';
3   let min = '';
4   let num = '';
5   for (let i = 0; i < str.length; i++) {
6     let digit = str[i];
7
8     if (digit !== ' ') {
9       if (str[i + 1] === ' ' || str[i + 1] === undefined) {
10        num = num + digit;
11        if (max === '' || min === '') {
12          max = num;
13          min = num;
14        }
15        if (Number(num) >= max) {
16          max = Number(num);
17        }
18        if (Number(num) <= min) {
19          min = Number(num);
20        }
21        num = '';
22      } else if (str[i + 1] !== ' ') {
23        num = num + digit;
24      }
25    }
26  }
27  return `${max} ${min}`
}
```

Sample Tests

```
1 const chai = require("chai");
2 const assert = chai.assert;
3 chai.config.truncateThreshold=0;
4
5 describe("Example tests", () => {
6   it("Test 1", () => {
7     assert.strictEqual(highAndLow("0 3 -5 42 -1 0 0 -9 4 7 4 -4"), "42 -9");
8   });
9   it("Test 2", () => {
10    assert.strictEqual(highAndLow("1 2 3"), "3 1");
11  });
12 });
13
```

Sixth task

Your task is to make a function that can take any non-negative integer as an argument and return it with its digits in descending order. Essentially, rearrange the digits to create the highest possible number.

Examples:

Input: 42145 Output: 54421

Input: 145263 Output: 654321

Input: 123456789 Output: 987654321

```
function descendingOrder(number){
  let n = String(number)
  if (n < 0 || n % 1 !== 0) {
    return console.log('Only positive integer numbers are allowed ');
  }
  else {
    let sort_num = '';
    for (let i = 0; i < n.length; i++) {
      let digit = n[i];
      let j = 0;

      while (j < sort_num.length && digit < sort_num[j]) {
        j++;
      }
      sort_num = sort_num.slice(0, j) + digit + sort_num.slice(j);
    }
    return Number(sort_num);
  }
}
```

7/10

Descending Order

☆ 2913

👤 523

📈 90% of 21,828

👤 93,582 of 237,947

👤 TastyOs

🚩 2 Issues Reported

Instructions

Output

Time: 816ms

Passed: 101

Failed: 0

Test Results:

Basic tests

Testing for fixed tests

Completed in 1ms

Random tests

descendingOrder(24398) should equal 98432

descendingOrder(215) should equal 521

descendingOrder(50) should equal 50

descendingOrder(5440922) should equal 9544220

descendingOrder(117328522) should equal 875322211

descendingOrder(70608) should equal 87600

descendingOrder(6) should equal 6

descendingOrder(4287) should equal 8742

descendingOrder(1873227) should equal 8773221

descendingOrder(1766499315) should equal 9976654311

descendingOrder(659770) should equal 977650

descendingOrder(272876) should equal 877622

descendingOrder(53120428) should equal 85432210

descendingOrder(7408430) should equal 8744300

descendingOrder(144) should equal 441

descendingOrder(22748) should equal 87422

JavaScript

Node v18.x

VIM

EMACS

🔍

Solution

```
1 • function descendingOrder(number){
2   let n = String(number)
3   if (n < 0 || n % 1 !== 0) {
4     return console.log('Only positive integer numbers are allowed ');
5   }
6   else {
7     let sort_num = '';
8     for (let i = 0; i < n.length; i++) {
9       let digit = n[i];
10      let j = 0;
11
12      while (j < sort_num.length && digit < sort_num[j]) {
13        j++;
14      }
15      sort_num = sort_num.slice(0, j) + digit + sort_num.slice(j);
16    }
17    return Number(sort_num);
18  }
19 }
```

Good Job! You may take your time to refactor/comment your solution. Submit when ready.

Sample Tests

```
1 const { assert } = require('chai')
2
3 describe("Basic tests", () => {
4   it("Testing for fixed tests", () => {
5     assert.strictEqual(descendingOrder(0), 0)
6     assert.strictEqual(descendingOrder(1), 1)
7     assert.strictEqual(descendingOrder(111), 111)
8     assert.strictEqual(descendingOrder(15), 51)
9     assert.strictEqual(descendingOrder(1021), 2110)
10    assert.strictEqual(descendingOrder(123456789), 987654321)
11  })
12 })
```