TS UZDEVUMI

*Define all functions as arrow functions like in the example.*

*Each function has a unique title.*

*Name functions in camelCase and with descriptive titles.*

*Define types for all arguments passed to the function and add a return type.*

EXAMPLE

**Write a function that takes two numbers (a and b) as argument**

**Subtract b from a**

**Return the result**

myFunction(1,2)

Expected

-1

myFunction(10,5)

Expected

5

myFunction(99,1)

Expected

98

ANSWER

const subtract = (*a*: number, *b*: number): number => a - b;

LONGER ANSWER EXAMPLE (FOR OTHER TASKS THAT HAVE MULTIPLE ACTIONS)

const subtract = (*a*: number, *b*: number): number => {

return a - b;

};

VERIFY ANSWER

console.log(subtract(1, 2)); // -1

console.log(subtract(10, 5)); // 5

console.log(subtract(99, 1)); // 98

**Write a function that takes two numbers (a and b) as argument**

**Sum a and b**

**Return the result**

myFunction(1,2)

Expected

3

myFunction(1,10)

Expected

11

myFunction(99,1)

Expected

100

**Write a function that takes a value as argument**

**Return the type of the value**

myFunction(1)

Expected

'number'

myFunction(false)

Expected

'boolean'

myFunction({})

Expected

'object'

myFunction(null)

Expected

'object'

myFunction('string')

Expected

'string'

myFunction(['array'])

Expected

'object'

**Write a function that takes two values, say a and b, as arguments**

**Return true if the two values are equal and of the same type**

myFunction(2, 3)

Expected

false

myFunction(3, 3)

Expected

true

myFunction(1, '1')

Expected

false

myFunction('10', '10')

Expected

true

**Write a function that takes a string (a) and a number (n) as arguments**

**Return the nth character of 'a'**

myFunction('abcd',1)

Expected

'a'

myFunction('zyxbwpl',5)

Expected

'w'

myFunction('gfedcba',3)

Expected

'e'

**Write a function that takes a string (a) as argument**

**Remove the first 3 characters of a**

**Return the result**

myFunction('abcdefg')

Expected

'defg'

myFunction('1234')

Expected

'4'

myFunction('fgedcba')

Expected

'dcba'

**Write a function that takes a string as argument**

**Extract the last 3 characters from the string**

**Return the result**

myFunction('abcdefg')

Expected

'efg'

myFunction('1234')

Expected

'234'

myFunction('fgedcba')

Expected

'cba'

**Write a function that takes a string (a) as argument**

**Get the first 3 characters of a**

**Return the result**

myFunction('abcdefg')

Expected

'abc'

myFunction('1234')

Expected

'123'

myFunction('fgedcba')

Expected

'fge'

**Write a function that takes a string (a) as argument**

**Extract the first half a**

**Return the result**

myFunction('abcdefgh')

Expected

'abcd'

myFunction('1234')

Expected

'12'

myFunction('gedcba')

Expected

'ged'

**Write a function that takes a string (a) as argument**

**Remove the last 3 characters of a**

**Return the result**

myFunction('abcdefg')

Expected

'abcd'

myFunction('1234')

Expected

'1'

myFunction('fgedcba')

Expected

'fged'

**Write a function that takes two numbers (a and b) as argument**

**Return b percent of a**

myFunction(100,50)

Expected

50

myFunction(10,1)

Expected

0.1

myFunction(500,25)

Expected

125

**Write a function that takes 6 values (a,b,c,d,e,f) as arguments**

**Sum a and b**

**Then substract by c**

**Then multiply by d and divide by e**

**Finally raise to the power of f and return the result**

**Tip: mind the order**

myFunction(6,5,4,3,2,1)

Expected

10.5

myFunction(6,2,1,4,2,3)

Expected

2744

myFunction(2,3,6,4,2,3)

Expected

-8

**Write a function that takes a number as argument**

**If the number is even, return true**

**Otherwise, return false**

myFunction(10)

Expected

true

myFunction(-4)

Expected

true

myFunction(5)

Expected

false

myFunction(-111)

Expected

false

**Write a function that takes two strings (a and b) as arguments**

**Return the number of times a occurs in b**

myFunction('m', 'how many times does the character occur in this sentence?')

Expected

2

myFunction('h', 'how many times does the character occur in this sentence?')

Expected

4

myFunction('?', 'how many times does the character occur in this sentence?')

Expected

1

myFunction('z', 'how many times does the character occur in this sentence?')

Expected

0

**Write a function that takes a number (a) as argument**

**If a is a whole number (has no decimal place), return true**

**Otherwise, return false**

myFunction(4)

Expected

true

myFunction(1.123)

Expected

false

myFunction(1048)

Expected

true

myFunction(10.48)

Expected

false

**Write a function that takes two numbers (a and b) as arguments**

**If a is smaller than b, divide a by b**

**Otherwise, multiply both numbers**

**Return the resulting value**

myFunction(10, 100)

Expected

0.1

myFunction(90, 45)

Expected

4050

myFunction(8, 20)

Expected

0.4

myFunction(2, 0.5)

Expected

1

**Write a function that takes two strings (a and b) as arguments**

**If a contains b, append b to the beginning of a**

**If not, append it to the end**

**Return the concatenation**

myFunction('cheese', 'cake')

Expected

'cheesecake'

myFunction('lips', 's')

Expected

'slips'

myFunction('Java', 'script')

Expected

'Javascript'

myFunction(' think, therefore I am', 'I')

Expected

'I think, therefore I am'

**Write a function that takes a number (a) as argument**

**Round a to the 2nd digit after the comma**

**Return the rounded number**

myFunction(2.12397)

Expected

2.12

myFunction(3.136)

Expected

3.14

myFunction(1.12397)

Expected

1.12

myFunction(26.1379)

Expected

26.14

**Write a function that takes a number (a) as argument**

**Split a into its individual digits and return them in an array**

**Tip: you might want to change the type of the number for the splitting**

myFunction(10)

Expected

[1,0]

myFunction(931)

Expected

[9,3,1]

myFunction(193278)

Expected

[1,9,3,2,7,8]

**It seems like something happened to these strings**

**Can you figure out how to clear up the chaos?**

**Write a function that joins these strings together such that they form the following words:**

**'Javascript', 'Countryside', and 'Downtown'**

**You might want to apply basic JS string methods such as replace(), split(), slice() etc.**

myFunction('java', 'tpi%rcs')

Expected

'Javascript'

myFunction('c%ountry', 'edis')

Expected

'Countryside'

myFunction('down', 'nw%ot')

Expected

'Downtown'

**This challenge is a little bit more complex**

**Write a function that takes a number (a) as argument**

**If a is prime, return a**

**If not, return the next higher prime number**

myFunction(38)

Expected

41

myFunction(7)

Expected

7

myFunction(115)

Expected

127

myFunction(2000)

Expected

2003

**Write a function that takes two numbers, say x and y, as arguments**

**Check if x is divisible by y**

**If yes, return x**

**If not, return the next higher natural number that is divisible by y**

myFunction(1, 23)

Expected

23

myFunction(23, 23)

Expected

23

myFunction(7, 3)

Expected

9

myFunction(-5, 7)

Expected

0

**Write a function that takes two strings (a and b) as arguments**

**Beginning at the end of 'a', insert 'b' after every 3rd character of 'a'**

**Return the resulting string**

myFunction('1234567','.')

Expected

'1.234.567'

myFunction('abcde','$')

Expected

'ab$cde'

myFunction('zxyzxyzxyzxyzxyz','w')

Expected

'zwxyzwxyzwxyzwxyzwxyz'

**Write a function that takes a string as argument**

**As it is, the string has no meaning**

**Increment each letter to the next letter in the alphabet**

**Return the correct word**

myFunction('bnchmf')

Expected

'coding'

myFunction('bgddrd')

Expected

'cheese'

myFunction('sdrshmf')

Expected

'testing'

**Write a function that takes an array (a) and a value (n) as argument**

**Return the nth element of 'a'**

myFunction([1,2,3,4,5],3)

Expected

3

myFunction([10,9,8,7,6],5)

Expected

6

myFunction([7,2,1,6,3],1)

Expected

7

**Write a function that takes an array (a) as argument**

**Remove the first 3 elements of 'a'**

**Return the result**

myFunction([1,2,3,4])

Expected

[4]

myFunction([5,4,3,2,1,0])

Expected

[2,1,0]

myFunction([99,1,1])

Expected

[]

**Write a function that takes an array (a) as argument**

**Extract the last 3 elements of a**

**Return the resulting array**

myFunction([1,2,3,4])

Expected

[2,3,4]

myFunction([5,4,3,2,1,0])

Expected

[2,1,0]

myFunction([99,1,1])

Expected

[99,1,1]

**Write a function that takes an array (a) as argument**

**Extract the first 3 elements of a**

**Return the resulting array**

myFunction([1,2,3,4])

Expected

[1,2,3]

myFunction([5,4,3,2,1,0])

Expected

[5,4,3]

myFunction([99,1,1])

Expected

[99,1,1]

**Write a function that takes an array (a) and a number (n) as arguments**

**It should return the last n elements of a**

myFunction([1, 2, 3, 4, 5], 2)

Expected

[ 4, 5 ]

myFunction([1, 2, 3], 6)

Expected

[ 1, 2, 3 ]

myFunction([1, 2, 3, 4, 5, 6, 7, 8], 3)

Expected

[ 6, 7, 8 ]

**Write a function that takes an array (a) and a value (b) as argument**

**The function should clean a from all occurrences of b**

**Return the filtered array**

myFunction([1,2,3], 2)

Expected

[1, 3]

myFunction([1,2,'2'], '2')

Expected

[1, 2]

myFunction([false,'2',1], false)

Expected

['2', 1]

myFunction([1,2,'2',1], 1)

Expected

[2, '2']

**Write a function that takes an array (a) as argument**

**Return the number of elements in a**

myFunction([1,2,2,4])

Expected

4

myFunction([9,9,9])

Expected

3

myFunction([4,3,2,1,0])

Expected

5

**Write a function that takes an array of numbers as argument**

**Return the number of negative values in the array**

myFunction([1,-2,2,-4])

Expected

2

myFunction([0,9,1])

Expected

0

myFunction([4,-3,2,1,0])

Expected

1

**Write a function that takes an array of numbers as argument**

**It should return an array with the numbers sorted in descending order**

myFunction([1,3,2])

Expected

[3,2,1]

myFunction([4,2,3,1])

Expected

[4,3,2,1]

**Write a function that takes an array of strings as argument**

**Sort the array elements alphabetically**

**Return the result**

myFunction(['b', 'c', 'd', 'a'])

Expected

['a', 'b', 'c', 'd']

myFunction(['z', 'c', 'd', 'a', 'y', 'a', 'w'])

Expected

['a', 'a', 'c', 'd', 'w', 'y', 'z']

**Write a function that takes an array of numbers as argument**

**It should return the average of the numbers**

myFunction([10,100,40])

Expected

50

myFunction([10,100,1000])

Expected

370

myFunction([-50,0,50,200])

Expected

50

**Write a function that takes an array of strings as argument**

**Return the longest string**

myFunction(['help', 'me'])

Expected

'help'

myFunction(['I', 'need', 'candy'])

Expected

'candy'

**Write a function that takes an array as argument**

**It should return true if all elements in the array are equal**

**It should return false otherwise**

myFunction([true, true, true, true])

Expected

true

myFunction(['test', 'test', 'test'])

Expected

true

myFunction([1,1,1,2])

Expected

false

myFunction(['10',10,10,10])

Expected

false

**Write a function that takes arguments an arbitrary number of arrays**

**It should return an array containing the values of all arrays**

myFunction([1, 2, 3], [4, 5, 6])

Expected

[1, 2, 3, 4, 5, 6]

myFunction(['a', 'b', 'c'], [4, 5, 6])

Expected

['a', 'b', 'c', 4, 5, 6]

myFunction([true, true], [1, 2], ['a', 'b'])

Expected

[true, true, 1, 2, 'a', 'b']

**Write a function that takes an array of objects as argument**

**Sort the array by property b in ascending order**

**Return the sorted array**

myFunction([{a:1,b:2},{a:5,b:4}])

Expected

[{a:1,b:2},{a:5,b:4}]

myFunction([{a:2,b:10},{a:5,b:4}])

Expected

[{a:5,b:4},{a:2,b:10}]

myFunction([{a:1,b:7},{a:2,b:1}])

Expected

[{a:2,b:1},{a:1,b:7}]

**Write a function that takes two arrays as arguments**

**Merge both arrays and remove duplicate values**

**Sort the merge result in ascending order**

**Return the resulting array**

myFunction([1, 2, 3], [3, 4, 5])

Expected

[ 1, 2, 3, 4, 5 ]

myFunction([-10, 22, 333, 42], [-11, 5, 22, 41, 42])

Expected

[ -11, -10, 5, 22, 41, 42, 333]

**Write a function that takes an array (a) and a number (b) as arguments**

**Sum up all array elements with a value greater than b**

**Return the sum**

myFunction([1, 2, 3, 4, 5, 6, 7], 2)

Expected

25

myFunction([-10, -11, -3, 1, -4], -3)

Expected

1

myFunction([78, 99, 100, 101, 401], 99)

Expected

602

**Write a function that takes two numbers (min and max) as arguments**

**Return an array of numbers in the range min to max**

myFunction(2, 10)

Expected

[2, 3, 4, 5, 6, 7, 8, 9, 10]

myFunction(1, 3)

Expected

[1, 2, 3]

myFunction(-5, 5)

Expected

[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5]

myFunction(2, 7)

Expected

[2, 3, 4, 5, 6, 7]

**Write a function that takes an array of strings as argument**

**Group those strings by their first letter**

**Return an object that contains properties with keys representing first letters**

**The values should be arrays of strings containing only the corresponding strings**

**For example, the array ['Alf', 'Alice', 'Ben'] should be transformed to**

**{ a: ['Alf', 'Alice'], b: ['Ben']}**

myFunction(['Alf', 'Alice', 'Ben'])

Expected

{ a: ['Alf', 'Alice'], b: ['Ben']}

myFunction(['Ant', 'Bear', 'Bird'])

Expected

{ a: ['Ant'], b: ['Bear', 'Bird']}

myFunction(['Berlin', 'Paris', 'Prague'])

Expected

{ b: ['Berlin'], p: ['Paris', 'Prague']}

**Write a function that takes an array with arbitrary elements and a number as arguments**

**Return a new array, the first element should be either the given number itself**

**or zero if the number is smaller than 6**

**The other elements should be the elements of the original array**

**Try not to mutate the original array**

myFunction([1,2,3], 6)

Expected

[6,1,2,3]

myFunction(['a','b'], 2)

Expected

[0,'a','b']

myFunction([null,false], 11)

Expected

[11,null,false]

**Write a function that takes an array (a) and a value (n) as arguments**

**Save every nth element in a new array**

**Return the new array**

myFunction([1,2,3,4,5,6,7,8,9,10],3)

Expected

[3,6,9]

myFunction([10,9,8,7,6,5,4,3,2,1],5)

Expected

[6,1]

myFunction([7,2,1,6,3,4,5,8,9,10],2)

Expected

[2,6,4,8,10]

**Write a function that takes an object with two properties as argument**

**It should return the value of the property with key country**

myFunction({ continent: 'Asia', country: 'Japan'})

Expected

'Japan'

myFunction({ country: 'Sweden', continent: 'Europe'})

Expected

'Sweden'

**Write a function that takes an object with two properties as argument**

**It should return the value of the property with key 'prop-2'**

**Tip: you might want to use the square brackets property accessor**

myFunction({ one: 1, 'prop-2': 2})

Expected

2

myFunction({ 'prop-2': 'two', prop: 'test'})

Expected

'two'

**Write a function that takes an object with two properties and a string as arguments**

**It should return the value of the property with key equal to the value of the string**

myFunction({ continent: 'Asia', country: 'Japan'}, 'continent')

Expected

'Asia'

myFunction({ country: 'Sweden', continent: 'Europe'}, 'country')

Expected

'Sweden'

**Write a function that takes an object (a) and a string (b) as argument**

**Return true if a has a property with key b**

**Return false otherwise**

myFunction({a:1,b:2,c:3},'b')

Expected

true

myFunction({x:'a',y:'b',z:'c'},'a')

Expected

false

myFunction({x:'a',y:'b',z:'c'},'z')

Expected

true

**Write a function that a string (a) as argument**

**Create an object that has a property with key 'key' and a value of a**

**Return the object**

myFunction('a')

Expected

{key:'a'}

myFunction('z')

Expected

{key:'z'}

myFunction('b')

Expected

{key:'b'}

**Write a function that takes two strings (a and b) as arguments**

**Create an object that has a property with key 'a' and a value of 'b'**

**Return the object**

myFunction('a','b')

Expected

{a:'b'}

myFunction('z','x')

Expected

{z:'x'}

myFunction('b','w')

Expected

{b:'w'}

**Write a function that takes two arrays (a and b) as arguments**

**Create an object that has properties with keys 'a' and corresponding values 'b'**

**Return the object**

myFunction(['a','b','c'],[1,2,3])

Expected

{a:1,b:2,c:3}

myFunction(['w','x','y','z'],[10,9,5,2])

Expected

{w:10,x:9,y:5,z:2}

myFunction([1,'b'],['a',2])

Expected

{1:'a',b:2}

**Write a function that takes an object (a) as argument**

**Return an array with all object keys**

myFunction({a:1,b:2,c:3})

Expected

['a','b','c']

myFunction({j:9,i:2,x:3,z:4})

Expected

['j','i','x','z']

myFunction({w:15,x:22,y:13})

Expected

['w','x','y']

**Write a function that takes an object (a) as argument**

**Return the sum of all object values**

myFunction({a:1,b:2,c:3})

Expected

6

myFunction({j:9,i:2,x:3,z:4})

Expected

18

myFunction({w:15,x:22,y:13})

Expected

50

**Write a function that takes an object as argument**

**It should return an object with all original object properties**

**except for the property with key 'b'**

myFunction({ a: 1, b: 7, c: 3 })

Expected

{ a: 1, c: 3 }

myFunction({ b: 0, a: 7, d: 8 })

Expected

{ a: 7, d: 8 }

myFunction({ e: 6, f: 4, b: 5, a: 3 })

Expected

{ e: 6, f: 4, a: 3 }

**Write a function that takes two objects as arguments**

**Unfortunately, the property 'b' in the second object has the wrong key**

**should be named 'd' instead**

**Merge both objects and correct the wrong property name**

**Return the resulting object**

**It should have the properties 'a', 'b', 'c', 'd', and 'e'**

myFunction({ a: 1, b: 2 }, { c: 3, b: 4, e: 5 })

Expected

{ a: 1, b: 2, c: 3, e: 5, d: 4}

myFunction({ a: 5, b: 4 }, { c: 3, b: 1, e: 2 })

Expected

{ a: 5, b: 4, c: 3, e: 2, d: 1}

**Write a function that takes an object (a) and a number (b) as arguments**

**Multiply all values of 'a' by 'b'**

**Return the resulting object**

myFunction({a:1,b:2,c:3},3)

Expected

{a:3,b:6,c:9}

myFunction({j:9,i:2,x:3,z:4},10)

Expected

{j:90,i:20,x:30,z:40}

myFunction({w:15,x:22,y:13},6)

Expected

{w:90,x:132,y:78}

**Write a function that takes an object as argument**

**Somehow, the properties and keys of the object got mixed up**

**Swap the Javascript object's key with its values and return the resulting object**

myFunction({z:'a',y:'b',x:'c',w:'d'})

Expected

{a:'z',b:'y',c:'x',d:'w'}

myFunction({2:'a',4:'b',6:'c',8:'d'})

Expected

{a:'2',b:'4',c:'6',d:'8'}

myFunction({a:1,z:24})

Expected

{1:'a',24:'z'}

**Write a function that takes an object as argument**

**Some of the property values contain empty strings**

**Replace empty strings and strings that contain only whitespace with null values**

**Return the resulting object**

myFunction({ a: 'a', b: 'b', c: '' })

Expected

{ a: 'a', b: 'b', c: null }

myFunction({ a: '', b: 'b', c: ' ', d: 'd' })

Expected

{ a: null, b: 'b', c: null, d: 'd' }

myFunction({ a: 'a', b: 'b ', c: ' ', d: '' })

Expected

{ a: 'a', b: 'b ', c: null, d: null }

**Write a function that takes an object as argument containing properties with personal information**

**Extract firstName, lastName, size, and weight if available**

**If size or weight is given transform the value to a string**

**Attach the unit cm to the size**

**Attach the unit kg to the weight**

**Return a new object with all available properties that we are interested in**

myFunction({fn: 'Lisa', ln: 'Müller', age: 17, size: 175, weight: 67})

Expected

{fn: 'Lisa', ln: 'Müller', size: '175cm', weight: '67kg'}

myFunction({fn: 'Martin', ln: 'Harper', age: 26, email: 'martin.harper@test.de', weight: 102})

Expected

{fn: 'Martin', ln: 'Harper', weight: '102kg'}

myFunction({fn: 'Andrew', ln: 'Harper', age: 81, size: 175, weight: 71})

Expected

{fn: 'Andrew', ln: 'Harper', size: '175cm', weight: '71kg'}

myFunction({fn: 'Matthew', ln: 'Müller', age: 19, email: 'matthew@mueller.de'})

Expected

{fn: 'Matthew', ln: 'Müller'}

**Write a function that takes an array of objects and a string as arguments**

**Add a property with key 'continent' and value equal to the string to each of the objects**

**Return the new array of objects**

**Tip: try not to mutate the original array**

myFunction([{ city: 'Tokyo', country: 'Japan' }, { city: 'Bangkok', country: 'Thailand' }], 'Asia')

Expected

[{ city: 'Tokyo', country: 'Japan', continent: 'Asia' }, { city: 'Bangkok', country: 'Thailand', continent: 'Asia' }]

myFunction([{ city: 'Stockholm', country: 'Sweden' }, { city: 'Paris', country: 'France' }], 'Europe')

Expected

[{ city: 'Stockholm', country: 'Sweden', continent: 'Europe' }, { city: 'Paris', country: 'France', continent: 'Europe' }]

**Write a function that takes an array of numbers as argument**

**Convert the array to an object**

**It should have a key for each unique value of the array**

**The corresponding object value should be the number of times the key occurs within the array**

myFunction([1,2,2,3])

Expected

{1:1,2:2,3:1}

myFunction([9,9,9,99])

Expected

{9:3,99:1}

myFunction([4,3,2,1])

Expected

{1:1,2:1,3:1,4:1}

**Write a function that takes two date instances as arguments**

**It should return true if the dates are equal**

**It should return false otherwise**

myFunction(new Date('2000/01/01 08:00:00'), new Date('2000/01/01 08:45:00'))

Expected

false

myFunction(new Date('2000/01/01 08:00:00'), new Date('2000/01/01 08:00:00'))

Expected

true

myFunction(new Date('2001/01/01 08:00:00'), new Date('2000/01/01 08:00:00'))

Expected

false

**Write a function that takes two date instances as argument**

**It should return the number of days that lies between those dates**

myFunction(new Date('2020-06-11'), new Date('2020-06-01'))

Expected

10

myFunction(new Date('2000-01-01'), new Date('2020-06-01'))

Expected

7457

**Write a function that takes two date instances as argument**

**It should return true if they fall on the exact same day**

**It should return false otherwise**

myFunction(new Date('2000/01/01'), new Date('2000/01/01'))

Expected

true

myFunction(new Date('2000/01/01'), new Date('2000/01/02'))

Expected

false

myFunction(new Date('2001/01/01'), new Date('2000/01/01'))

Expected

false

myFunction(new Date('2000/11/01'), new Date('2000/01/01'))

Expected

false

SPREAD OPERATOR

*Use spread operator in all tasks*

**Write a function that takes two number arrays as parameters**

**and return an array which contains elements from both**

**arrays**

myFunction([1, 2], [3, 4])

Expected

[1, 2, 3, 4]

myFunction([1, 2], [3, 4, 5, 6])

Expected

[1, 2, 3, 4, 5, 6]

**Write a function that takes an array and a string as parameters**

**and return an array which contains all elements from the given array**

**and the given string as the last element**

myFunction(['Apple', 'Orange', 'Banana'], 'Kiwi');

Expected

['Apple', 'Orange', 'Banana', 'Kiwi']

**Write a function that takes an array and a string as parameters**

**and return an array which contains all elements from the given array**

**and the given string as the first element**

myFunction(['Apple', 'Orange', 'Banana'], 'Kiwi');

Expected

['Kiwi', 'Apple', 'Orange', 'Banana']

**Write a function that takes two objects as parameters**

**and return an object which contains properties from both**

**objects**

myFunction({ a:1, b:2 }, { c:3, d:4 })

Expected

{ a:1, b:2, c:3, d:4 }

myFunction({ a:1, b:2 }, { c:3, d:4, e:5, f:6 })

Expected

{ a:1, b:2, c:3, d:4, e:5, f:6 }

**Write a function that takes an object and a string as parameters**

**and return an object which contains properties from the given object**

**and a new property favoriteMovie with the value equal to the given string**

myFunction({ eyeColor: 'green', age: 10 }, 'Garfield')

Expected

{ eyeColor: 'green', age: 10, favoriteMovie: 'Garfield' }

myFunction({ eyeColor: 'blue', age: 15 }, 'Twilight')

Expected

{ eyeColor: 'blue', age: 15, favoriteMovie: 'Twilight' }