مربع هر رقم از ورودی (1

```
function squareDigits(n) {
	return +[...String(n)].map(x => x*x).join('');
}
```

```
const squareDigits = n => +(''+n).replace(/./g, v => v*v));
```

```
const squareDigits = n => +n.toString().split('').map(x=>x*x).join('');
```

```
function squareDigits(n) {
	return +[...String(n)].map(x => Math.pow(+x, 2)).join('');
}
```

```
return parseInt(Array.from(n.toString(), digit => digit * digit).join(''));
```

```
function squareDigits(n) {
	total='';
	n= n.toString()
	for(i=0;i<n.length;i++){
		total+=Math.pow(n[i],2)
	}
	return parseInt(total)
}
```

```
function squareDigits(n) {
	let arr = String(n).split('')
	return +arr.map(num => Math.pow(+num, 2)).join('')
}
```

```
function squareDigits(n) {
	n = n.toString().split('').map(num => Number(num) * Number(num)).join('');
	return Number(n);
```

رنگ خانه های شطرنج (2

```
const chessBoard = ([file, rank]) => (
        (file.charCodeAt() % 2 === rank % 2) ? 'black' : 'white'
);
```

```
function chessBoard (pole) {
  return ["black", "white"][(pole.charCodeAt(0) + +pole[1]) % 2];
}
```

```
const chessBoard = p =>(p.charCodeAt()+ +p[1]) % 2 ? "white" : "black";
```

```
function chessBoard (pole) {
    let bukva = pole[0].toUpperCase();
    let cifra = parseInt(pole[1]);

    if (bukva == 'A' || bukva == 'C' || bukva == 'E' || bukva == 'G') {
        if (cifra % 2 == 0) {
            return 'white';
        } else {
            return 'black';
        }
    } else {
        if (cifra % 2 == 0) {
            return 'black';
        } else {
            return 'white';
        }
    }
}
```

```
const chessBoard=p=>(parseInt(p,35))%2==0?'white':'black'
```

اعداد صحیح موجود در بازه ورودی از کاربر (3

```
 function reversibleInclusiveList(s,e) {
        var ans = []
        for (let i = Math.min(s,e); i <= Math.max(s,e); i++) { ans.push(i) }
        return s > e ? ans.reverse() : ans
}
```

```
const reversibleInclusiveList=(s, e) =>[...Array(Math.abs(s-e)+1)].map((_, i)=> s > e ? s - i : s +
i);
```

```
function reversibleInclusiveList(start, end) {
        const loop = end > start ? 'i <= end; i++' :
                'i >= end; i--';
        let result = [];
        eval(`for (let i = start; ${loop}) result.push(i);`)
        return result;
}
```

```
function reversibleInclusiveList(start, end) {
        return Array.from({length: Math.abs(end-start)+1}, _=> start>end ? start-- : start++)
}
```

```
function reversibleInclusiveList(start, end) {
  let res = [];
  if (start > end) for (let i = start; i >= end; i--) res.push(i);
  else for (let i = start; i <= end; i++) res.push(i);
  return res;
}
```

```
const reversibleInclusiveList = (s, e, r=0) => {
  [s, e, r] = s > e ? [e, s, 1] : [s, e, r]
  let v = Array.from({length: e + 1 - s}, () => s++)
  return r ? v.reverse() : v
}
```

چند پارامتر تکراری داریم؟ (4

```
function equal(a, b, c) {
	const size = (new Set([a, b, c])).size;
	return size === 3 ? 0 : 4 - size;
}
```

```
function equal(a, b, c) {
	 if (a === b && a === c) {
		return 3;}
	 if (a === b || a === c || b === c) {
		return 2;}
			return 0;
}
```

```
function equal(a, b, c) {
	return [...arguments].filter((e, i, a) => {
		return a.filter((x, idx) => idx != i).includes(e)
	}).length;
}
```

```
const equal = (a, b, c) => {
	var z = 4 - [... new Set([a, b, c])].length
	return z == 1 ? 0 : z;
}
```

```
function equal(a, b, c) {
	if (a !== b && b !== c && a !== c) {
		return 0;
	} else if (a === b && b === c && a === c) {
		return 3;
	} else {
		return 2;
	}
}
```

بازی سنگ، کاغذ، قیچی (5

```
function rps(p1, p2) {
  let w = {
    Rock: "Scissors",
    Scissors: "Paper",
    Paper: "Rock"
  }
  if (p1 === p2) return "It's a draw"
        return `The winner is ${w[p1] === p2 ? 'p1' : 'p2'}`
}
```

```
const rps = (p1, p2) => {
        dic = {Rock: 'Scissors', Scissors: 'Paper', Paper: 'Rock'};
        return p1==p2? "It's a draw" : `The winner is p${2-(dic[p1]==p2)}`;
};
```

```
function rps(p1, p2) {
        const wins = ['RockScissors', 'PaperRock', 'ScissorsPaper'];
        return p1 === p2 ? "It's a draw" : `The winner is ${wins.includes(p1 + p2) ? 'p1' : 'p2'}`;
}
```

```
function rps(p1, p2) {
        let obj = {
                Rock: 'Scissors',
                Scissors: 'Paper',
                Paper: 'Rock'
        }
        return p1 === p2 ? "It's a draw" : obj[p1] === p2 ? "The winner is p1" : "The winner is p2";
}
```

آرایه ای از مضارب عدد ورودی (6

```
const arrayOfMultiples = (num, length) => Array.from({ length }, (_, i) => num * (i + 1));

function arrayOfMultiples (num, length) {
        var result = []
   for (let i = 1; i <= length; i++) {
      result.push(num*i)

   }
   return result
}
---------------------------------------------------------------------------------------------------
const arrayOfMultiples = (num, length) => {
        return Array.from({length: length}, (_, i) => num * (i + 1));
}
---------------------------------------------------------------------------------------------------
function arrayOfMultiples (num, length) {
        return [...Array(length)].map((_, i) => num * (i + 1))
}
---------------------------------------------------------------------------------------------------
function arrayOfMultiples (num, length) {
        let output = []
        let multiple = 0
        for(i = 1; i < length + 1; i++) {
                multiple = num * i
                output.push(multiple)
        }
        return output
}
---------------------------------------------------------------------------------------------------
arrayOfMultiples = (n, l)  => [...Array(l).keys()].map(x => (x + 1) * n)
---------------------------------------------------------------------------------------------------
```

استخراج اعضای آبجکت (7

```
let names = []

let users = [
  { name: "John", email: "john@example.com" },
  { name: "Jason", email: "jason@example.com" },
  { name: "Jeremy", email: "jeremy@example.com" },
  { name: "Jacob", email: "jacob@example.com" }
]

const str = `
        for( let {name} of users ) {
                        names.push(name)
        }
`
```

```
let names = []

let users = [
  { name: "John", email: "john@example.com" },
  { name: "Jason", email: "jason@example.com" },
  { name: "Jeremy", email: "jeremy@example.com" },
  { name: "Jacob", email: "jacob@example.com" }
]

const str = `
        for({ name } of users) {
                        names.push(name)
        }`
```

استخراج اعضای آبجکت (8

```
function keysAndValues(obj) {
  var keys = Object.keys(obj);
  return [keys, keys.map( key => obj[key] )];
}
```

```
function keysAndValues(obj) {
        const arrs = [Object.keys(obj), Object.values(obj)];
        return arrs;
}
```

```
function keysAndValues(obj) {
  return [Object.keys(obj), Object.keys(obj).map(x => obj[x])];
}
```

```
function keysAndValues(obj) {
  var k = [];
  var o = [];
  for (var i in obj){
    k.push(i);
    o.push(obj[i]);
  }
  return [k,o];
}
```

```
const keysAndValues = obj=> [ Object.keys(obj), Object.values(obj)]
```

```
keysAndValues=o=>(f=Object,[f.keys(o),f.values(o)])
```

```
function keysAndValues(obj) {
        let keys = Object.keys(obj);
        let values = Object.values(obj);
        return [keys, values]
}
```

معدل نمرات دانش آموز (9

```
const sum = arr => arr.reduce((total, num) => total + num, 0);
```

```
const getStudentsWithNamesAndAvgNote = students =>
  students.map(student => ({
    name: student.name,
    avgNote: sum(student.notes) / student.notes.length || 0,
  }));
```

```
getStudentsWithNamesAndAvgNote=s=>s.map(({name,notes})=>({name,avgNote:eval(notes.join('+'))/notes.length||0}))
```

```
function getStudentsWithNamesAndAvgNote(students) {
  return students.map(item=> {
    const len = item.notes.length
    const avg = len ? (item.notes.reduce((acc, val)=> acc + val) / len) : 0
    return {name: item.name, avgNote: avg}
  })
```

```
const getStudentsWithNamesAndAvgNote= (students)=> {
 return students.map(student => student = {
          name: student.name, avgNote: student.notes.length === 0 ? 0 : student.notes.reduce(((a,b) => a+b),0)/student.notes.length});
 }
```

```
function getStudentsWithNamesAndAvgNote(students) {
        return students.map((item)=>( {name: item.name, avgNote:(!item.notes.length)?0:item.notes.reduce((sum,item)=>sum+item,0)/item.notes.length
}));
}
```

پیدا کردن اجداد و نوادگان با کد (10)

```
function generation(x,y) {
const elo = {
  '-3': { m: "great grandfather", f: "great grandmother" },
  "-2": { m: "grandfather", f: "grandmother" },
  "-1": { m: "father", f: "mother" },
  0: { m: "me!", f: "me!" },
  1: { m: "son", f: "daughter" },
  2: { m: "grandson", f: "granddaughter" },
  3: { m: "great grandson", f: "great granddaughter" },
};
        return elo[x][y]
}
```

```
const generation = (g, s) => ([,,'grand','great grand'][Math.abs(g)] || '') +
        [(s=='m'?'fa':'mo')+'ther','me!',s=='m'?'son':'daughter'][Math.sign(g) + 1];

function generation(x,y) {
        let f = ['great grandmother','grandmother','mother','me!','daughter','granddaughter','great
granddaughter'];
        let m = ['great grandfather','grandfather','father','me!','son','grandson','great grandson'];
        return y === 'm' ? m[3+x] : f[3+x];
```

```
function generation(x,y) {
        const generations = {
                "-3": ["great grandfather","great grandmother"],
                "-2": ["grandfather",          "grandmother"],
                "-1": ["father", "mother"],
                 0: ["me!", "me!"],
                 1: ["son", "daughter"],
                 2: ["grandson",      "granddaughter"],
                 3: ["great grandson",       "great granddaughter"]
        }
        return generations[x][y === "m" ? 0 : 1]
}
```