### 1) Write a function to transform array a to array b and print the elements of array b to the console.

let a = [2, 4, 6, 8, 9, 15]

let b = ['4', '16', '64']

Ans:

b.push.apply(b, a);

console.log(b);

### 2) Write a function to calculate the cumulative TTL of the following set of requests. (The provided answer is correct and should not be modified. )

let requests = [

{requestId: 'poiax',  startedAt: 1489744808, ttl: 8},

{requestId: 'kdfhd',  startedAt: 1489744803, ttl: 3},

{requestId: 'uqwyet', startedAt: 1489744806, ttl: 12},

{requestId: 'qewaz',  startedAt: 1489744810, ttl: 1}

]

let cumulativeTtl = 15

Ans:

var max = result = 0;

var min = requests[0].startedAt;

requests.forEach(function(row){

max = ((row.startedAt + row.ttl) > max)

? (row.startedAt + row.ttl)

: max;

min = (row.startedAt < min)

? row.startedAt

: min;

});

result = (max - min);

console.log((result == cumulativeTtl) ? true : false);

### 3) Read a text file containing a set of flat polygons represented as one polygon per line. Each line contains a comma-separated list of side lengths (for example “3,4,8,5,7”). Write a function to classify the set of polygons into four mutually exclusive subsets: triangles, rectangles, squares, and everything else. The union of all four subsets should be the original set of polygons. All the sides of the polygons are connected and the angles between them are irrelevant. Only consider the lengths.

Ans:

<script>

/\*\*

\* count a value how many times in an array

\*/

function countValueInArray(array, niddle) {

return array.filter(item => item == niddle).length;

}

/\*\*

\* read file

\*/

var fileText = [];

function readTextFile(file) {

var rawFile = new XMLHttpRequest();

rawFile.open("GET", file, false);

rawFile.onreadystatechange = function() {

if (rawFile.readyState === 4) {

if (rawFile.status === 200 || rawFile.status == 0) {

// when text file content json : [["1,3,1"], ["2,2,2,2"], ["2,4,2,4"], ["4,2,3"], ["3,4,2,6"]]

//fileText = JSON.parse(rawFile.responseText);

// for line separated file

fileText = rawFile.responseText.split('\n');

//alert(fileText);

}

}

}

rawFile.send(null);

}

/\*\*

\* this will check whether it can make a triangle

\*/

function checkTriangle(set) {

if (set.length != 3) {

return false;

}

var a = parseInt(set[0]);

var b = parseInt(set[1]);

var c = parseInt(set[2]);

if (((a + b) > c) && ((a + c) > b) && ((b + c) > a)) {

return true;

} else {

return false;

}

}

/\*\*

\* this will check whether it can make a square

\*/

function checkSquare(set) {

if (set.length != 4) {

return false;

}

var a = parseInt(set[0]);

var b = parseInt(set[1]);

var c = parseInt(set[2]);

var d = parseInt(set[3]);

var equal = set.reduce(function(sum, val) {

return val === set[0];

});

return equal;

}

/\*\*

\* this will check whether it can make rectangle or not

\*/

function checkRectangle(set) {

if (set.length != 4) {

return false;

}

var a = parseInt(set[0]);

var b = parseInt(set[1]);

var c = parseInt(set[2]);

var d = parseInt(set[3]);

if (((countValueInArray(set, a) == 2) && (countValueInArray(set, b) == 2)) || ((countValueInArray(set, c) == 2) && (countValueInArray(set, d) == 2)) || ((countValueInArray(set, a) == 2) && (countValueInArray(set, c) == 2)) || ((countValueInArray(set, a) == 2) && (countValueInArray(set, d) == 2)) || ((countValueInArray(set, b) == 2) && (countValueInArray(set, d) == 2)) || ((countValueInArray(set, b) == 2) && (countValueInArray(set, c) == 2))) {

return true;

} else {

return false;

}

}

// create polygons array

var polygons = [];

readTextFile('./polygons.txt');

//document.write(fileText.length);

fileText.forEach(function(p) {

polygons.push(p);

});

//document.write(polygons[0]);

var triangles = [];

var squares = [];

var rectangles = [];

var others = [];

var result;

// polygons loop of all sets

for (let i = 0; i < polygons.length; i++) {

var set;

set = polygons[i].split(',');

//document.writeln(set.length);

if (set.length == 3) {

result = checkTriangle(set);

if (result == true) {

triangles.push(set);

} else {

others.push(set);

}

} else if (set.length == 4) {

result = checkSquare(set);

if (result == true) {

squares.push(set);

//rectangles.push(set);

} else if (checkRectangle(set) == true) {

rectangles.push(set);

} else {

others.push(set);

}

} else {

others.push(set);

}

}

/\*let unionOfAll = new Set(triangles, squares, rectangles, others);

unionOfAll.forEach(function(set){

document.writeln(set);

});\*/

// as they are already mutually exclusive set/array

let unionOfAll = triangles.concat(squares.concat(triangles.concat(others)));

document.write("Original polygons txt file contents: <br>" + fileText.join("<br>") + "<br><hr><br>");

document.write("Mutually exclusive sets:-<br>");

document.write("Triangles: " + JSON.stringify(triangles) + "<br>");

document.write("Squares: " + JSON.stringify(squares) + "<br>");

document.write("Rectangles: " + JSON.stringify(rectangles) + "<br>");

document.write("Others: " + JSON.stringify(others) + "<br><hr><br>");

document.write("Union of all: " + JSON.stringify(unionOfAll) + "<br>");

</script>