1) **Que imprime el alert() y porque?**

```
var a = 1;
function f(){
        var a = 2;
        function n(){
                alert(a);
        }
        n();
}
                f();
```

2) Explique que pasa con este codigo

```
(
        function(){
                return alert;
        }
)()('Boo');
```

3) Que imprime o.name? Porque?

```
function F(){
        this.name = "f";
        function C(){
                return this;
        }
        return C();
}
var o = new F();
o.name;
```

4) Que imprime en la consola?

```
function C(){
        this.a = 1;
        return false;
}

console.log(typeof new C());
```

5) Que imprime este codigo?

```
(
        function() { var x = y = 5; }
)();
console.log(x);
console.log(y);
```

6) Definir una función llamada repeatify en el objeto String.

Ejemplo de uso:

```
console.log('texto'.repeatify(2));  //textotexto
```

7) Implementar un modelo de objetos donde exista un objeto Triangulo, y otro cuadrado que hereden de Forma. Donde forma tiene un metodo getArea, que cada uno implementa.

```
function calculateTotal(shapes){
        var total = 0;
        shapes.forEach(function(shape){
                total += shape.getArea();
        });
        return total;
}

var dummy = {};
dummy.getArea = function (){ return 1;}
calculateTotal([dummy,dummy,dummy]) == 3;
```

8) Analice este codigo, es correcto?

```
function Logger(base){
   // base...
        this.log = function log(message){ console.log(message);}
}

function User(){
        this.lastProcessDate = null;
        this.process = function(context, errorCallback, successCallback){
                //process
                return successCallback( {} );
        }
}
```

```
function UserService(context){
        this.logger = new Logger("UserService");
        this.context = context;
        this.processData = function(user){

                user.process(this.context, function(error){
                                        this.logger.log(error);
                                        if (error) return error;
                                },
                                function(result){
                                        this.logger.log(result.toString());
                                        user.lastProcessDate = new Date();
                                        return result;
                                });
        }
}
var context = {};
var service = new UserService(context);
var user = new User("Juan");
service.processData(user);
console.log(user.lastProcessDate);
```

9) Javascript no maneja namespaces. Como implementaria algun mecanismo tipo namespace para poder crear nuevas funciones dentro del mismo y evitar colisiones de nombres con otras librerias?

Implementar:

1) Implementar el juego Fizz Buzz, que dado un nro responda con la respuesta correcta.

Fizz buzz is a group word game for children to teach them about division.[1] Players take turns to count incrementally, replacing any number divisible by three with the word "fizz", and any number divisible by five with the word "buzz".

2) Implementar el siguiente ejercicio:

A zero-indexed array A consisting of N integers is given. An *equilibrium index* of this array is any integer P such that 0 ≤ P < N and the sum of elements of lower indices is equal to the sum of elements of higher indices, i.e.

A[0] + A[1] + ... + A[P−1] = A[P+1] + ... + A[N−2] + A[N−1].

Sum of zero elements is assumed to be equal to 0. This can happen if P = 0 or if P = N−1.

For example, consider the following array A consisting of N = 8 elements:

A[0] = -1   A[1] = 3   A[2] = -4   A[3] = 5   A[4] = 1   A[5] = -6   A[6] = 2   A[7] = 1

P = 1 is an equilibrium index of this array, because:

- A[0] = −1 = A[2] + A[3] + A[4] + A[5] + A[6] + A[7]

P = 3 is an equilibrium index of this array, because:

- A[0] + A[1] + A[2] = −2 = A[4] + A[5] + A[6] + A[7]

P = 7 is also an equilibrium index, because:

- A[0] + A[1] + A[2] + A[3] + A[4] + A[5] + A[6] = 0

and there are no elements with indices greater than 7.

P = 8 is not an equilibrium index, because it does not fulfill the condition 0 ≤ P < N.

Write a function: `function solution(A);`

that, given a zero-indexed array A consisting of N integers, returns any of its equilibrium indices. The function should return −1 if no equilibrium index exists.

For example, given array A shown above, the function may return 1, 3 or 7, as explained above.

Assume that:

- N is an integer within the range [0..100,000];
- each element of array A is an integer within the range [−2,147,483,648..2,147,483,647].

Complexity:

- expected worst-case time complexity is O(N);
- expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.