Práctica 10

DOCENTE	CARRERA	CURSO
MSc. Vicente Enrique Machaca Arceda	Escuela Profesional de Ingeniería de Software	Fundamentos de Lenguajes de Programación

PRÁCTICA	TEMA	DURACIÓN
10	Programación Funcional	2 horas

1. Datos de los estudiantes

- Grupo: 03
- Integrantes:
  - Jonathan Aguirre Soto

2. Ejercicios

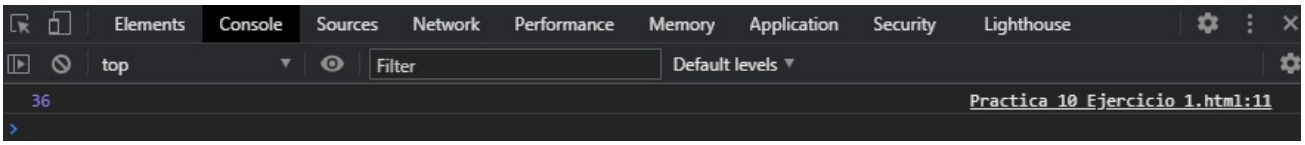
Implemente funciones en Javascript que:

1. Implemente una función llamada composedValue que tome como parametros la dos funciones (square and double) y un valor x. Luego esta función debe retornar el resultado de: (f ∘ g)(x); donde f y g son square y double respectivamente. **(2 puntos)**

```
function square (x) { return (x*x); }
function double (x) { return (x*2) ; }
function composedValue ( square , double , x ){
// ...
}
console .log ( composedValue ( square , double , 3 ) ); // 36
```

```
<html><body>
<script src = "ramda.min.js"> </script>
<script>
 function square (x) { return (x*x); }
 function double (x) { return (x*2) ; }
 function composedValue ( square, double, x ){
    composed = R.compose (square, double);
    variable1 = composed(x);
    return variable1;
 }
 console.log(composedValue(square, double, 3)) //36
</script>
</body> </html>
```

- Ejecución del programa.

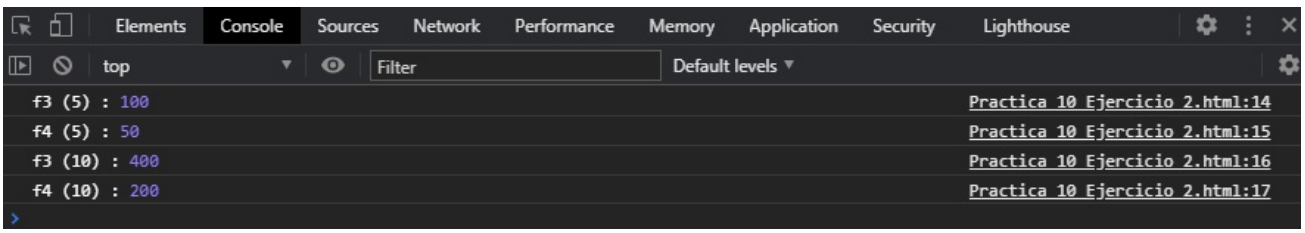


2. Implemente una función compose que tome como parametros dos funciones (f1 y f2) y luego retorne otra función que representa la función compuesta: (f1 ∘ f2)(x).**(2 puntos)**

```
function compose ( f1 , f2 ){
  // ...
  }
  var f3 = compose ( square , double ) ;
  var f4 = compose ( double , square ) ;
  console .log ("f3 (5) :", f3 (5) ); // 100
  console .log ("f4 (5) :", f4 (5) ); // 50
  console .log ("f3 (10) :", f3 (10) ); // 400
  console .log ("f4 (10) :", f4 (10) ); // 200
```

```
<html><body>
<script src = "ramda.min.js"> </script>
<script>

 function square (x) { return (x*x);}
 function double (x) { return (x*2);}

 function compose (square, double){
    return variable1 = R.compose(square, double);
 }

 var f3 = compose ( square , double );
 var f4 = compose ( double , square );
 console .log ("f3 (5) :", f3 (5) ); // 100
 console .log ("f4 (10) :", f4 (5) ); // 50
 console .log ("f3 (5) :", f3 (10) ); // 400
 console .log ("f4 (10) :", f4 (10) ); // 200

</script>
</body> </html>
```

- Ejecución del programa.

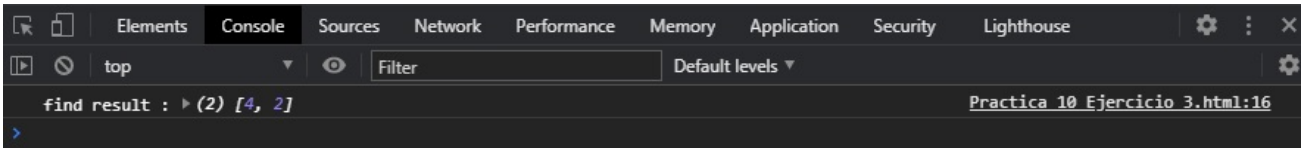


3. Implemente una función find que tome como parametros un array y una función f (condición). Luego, debe retornar otro vector que resulte de aplicar un filtro según f en cada elemento del vector. Solo los elementos que cumplan la condición (f) estarán presenten en el vector de resultado **(2 puntos)**

```
function isEven ( num ) { return ( num %2 == 0) ; }
  function find ( arr , f ){
  // ...
  }
  isEven (3) // false
  isEven (4) // true
  result = find ([1 , 3, 5, 4, 2] , isEven );
  console .log (" find result :", result ); // [4 , 2]
```

```
<html><body>
<script src = "ramda.min.js"> </script>
<script>
 function isEven (num) { return ( num %2 == 0);}
 function find (arr, isEven){

    let new_array = [];

    for(var i=0; i<arr.length; i++)
       if(isEven(arr[i]))
                        new_array.push(arr[i]);

    return new_array;
 };

       //console.log(isEven (3)); // false
 //console.log(isEven (4)); // true

 result = find ([1, 3, 5, 4, 2], isEven);
 console.log (" find result :", result ); // [4 , 2]

</script>
</body></html>
```

- Ejecución del programa.



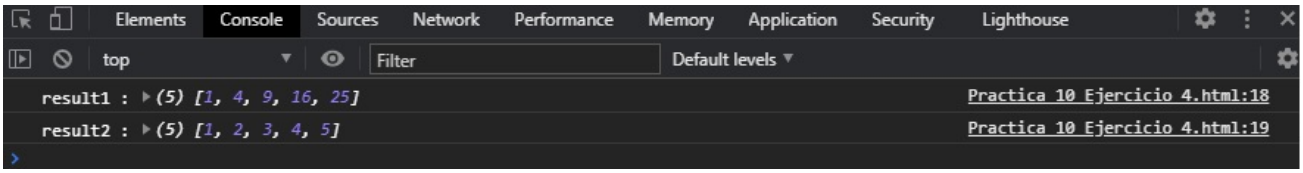
4. Implemente una función mymap que tome como parametros un array y una función f. Luego, debe retornar otro vector que resulte de aplicar la función f en cada elemento del vector. **(2 puntos)**

```
function mymap (arr , f){
  \\ ...
```

```
}
result1 = mymap ([1 , 2 , 3 , 4, 5] , square );
result2 = mymap ([1 , 4 , 9 , 16 , 25] , Math . sqrt );
console .log (" result1 :", result1 ); // [1 , 4, 9, 16 , 25]
console .log (" result2 :", result2 ); // [1 , 2, 3, 4, 5]
```

```
<html><body>
<script src = "ramda.min.js"> </script >
<script>
 function square (x) { return (x*x);}
 function mymap(arr, f){

    let new_array = [];

    for(var i=0; i<arr.length; i++)
       arr[i] = new_array.push(f(arr[i]));

    return new_array;
 };

 result1 = mymap([1 , 2 , 3 , 4, 5] , square);
 result2 = mymap([1 , 4 , 9 , 16 , 25] , Math.sqrt);

 console .log (" result1 :", result1 ); // [1 , 4, 9, 16 , 25]
 console .log (" result2 :", result2 ); // [1 , 2, 3, 4, 5]

</script>
</body></html>
```

- Ejecución del programa.

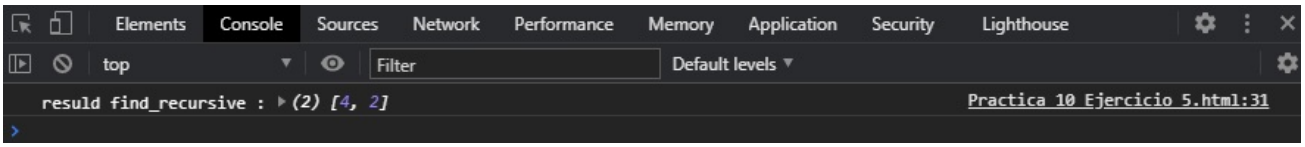


5. Implemente una función find recursive que realice lo mismo que la función find implementada anteriormente. Pero ahora, esta función debe respetar el paradigma funcional, es decir, debe ser una función pura y no debe contener ciclos. NOTA: Puede utilizar la función slice y concat. **(4 puntos)**

```
function find_recursive (arr , f){
  // ...
  }
  result = find_recursive ([1 , 3, 5, 4, 2] , isEven );
  console .log (" resuld find_recursive :", result ); // [4 , 2]
```

```
<html><body>
<script src = "ramda.min.js"> </script>
<script>

 var new_array = [];
 var i = 0; //va a estar en toda la funcin
 var aux = []; //variable auxiliar

 function isEven (num) { return (num %2 == 0);}
 function find_recursive(arr, isEven){

    if(i < arr.length){
       if(isEven(arr[i])){
          new_array = new_array.concat(arr[i]);
                //new_array = new_array.slice(arr[i]);
       };

       i = i + 1;
       find_recursive(arr, isEven);
    }

    if(i == arr.length){
       aux = new_array;
       new_array = [];
       i = 0;
    }

    return aux;
 };

 result = find_recursive ([1, 3, 5, 4, 2] , isEven);
 console.log(" resuld find_recursive :", result); // [4 , 2]

</script>
</body> </html>
```

- Ejecución del programa.



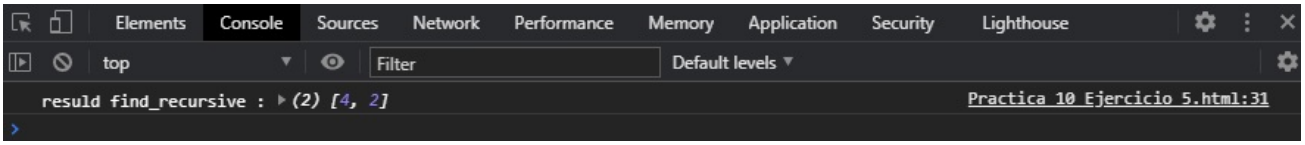
6. Implemente una función mymap recursive que realice lo mismo que la función mymap implementada anteriormente. Pero ahora, esta función debe respetar el paradigma funcional, es decir, debe ser una función pura y no debe contener ciclos. NOTA: Puede utilizar la función slice y concat. **(4 puntos)**

```
function mymap_recursive (arr , f){
  // ...
  }
  result1 = mymap_recursive ([1 , 2, 3 , 4 , 5] , square , result1 );
  result2 = mymap_recursive ([1 , 4, 9 , 16 , 25] , Math .sqrt , result2 );
  console .log (" mymap_recursive result1 :", result1 ); // [1 , 4, 9, 16 , 25]
  console .log (" mymap_recursive result2 :", result2 ); // [1 , 2, 3, 4, 5]
```

```
<html><body>
<script src = "ramda.min.js"> </script >
<script>

   var new_array = [];
   var i = 0; //va a estar en toda la funcin
   var aux = []; //variable auxiliar

 function square (x) { return (x*x);}
 function mymap_recursive(arr, f){

   if(i < arr.length){

      new_array = new_array.concat(f(arr[i]));
                    //new_array = new_array.slice(f(arr[i]));

      i=i+1;
      mymap_recursive(arr, f);
   }

   if(i == arr.length){
      aux = new_array;
      new_array = [];
      i = 0;
   }

   return aux;
 };

 result1 = mymap_recursive ([1 , 2, 3 , 4 , 5] , square);
 result2 = mymap_recursive ([1 , 4, 9 , 16 , 25] , Math.sqrt);
 console.log (" mymap_recursive result1 :", result1 ); // [1 , 4, 9, 16 , 25]
 console.log (" mymap_recursive result2 :", result2 ); // [1 , 2, 3, 4, 5]

</script>
```

```
</body> </html>
```

- Ejecución del programa.



7. Enlace de Github:
   https://github.com/Jona2010/Fundamentos-de-Lenguaje-de-la-Programaci-n