



Universidad La Salle

Ingeniería de Software

Fundamentos de Lenguajes de Programación

Karlo Pacha Curimayhua

Fecha: Octubre 25, 2022

Práctica 13

Ejercicios

1. Investigue el concepto de first class en Javascript y muestre una pequeña definición seguida ejemplos. (2 puntos)
 - (a) Son funciones que son tratadas como cualquier otra variable; estas pueden ser pasadas como argumento a otras funciones, pueden ser retornadas por otra función y pueden ser asignadas a una variable.

```
const Arithmetics = {  
  add: (a, b) => {  
    return `${a} + ${b} = ${a + b}`;  
  },  
  subtract: (a, b) => {  
    return `${a} - ${b} = ${a - b}`  
  },  
  multiply: (a, b) => {  
    return `${a} * ${b} = ${a * b}`  
  },  
  division: (a, b) => {  
    if (b !== 0) return `${a} / ${b} = ${a / b}`;  
    return 'Cannot Divide by Zero!!!';  
  }  
}
```

2. Describa la diferencia entre Currying and Partial Application. Incluya ejemplos. (2 puntos)

- (a) Currying: Una función que toma una función con múltiples parámetros como entrada y devuelve una función con exactamente un parámetro.
- (b) Partial Application: El proceso de aplicar una función a algunos de sus argumentos. La función aplicada parcialmente se devuelve para su uso posterior.

```
function partial(firstArgument, secondArgument) {  
  return function(thirdArgument, fourthArgument, fifthArgument) {  
    return firstArgument + secondArgument + thirdArgument + fourthArgument + fifthArgument;  
  }  
}
```

```
function curry(firstArgument) {  
  return function(secondArgument) {  
    return function(thirdArgument) {  
      return function(fourthArgument) {  
        return function(fifthArgument) {  
          return firstArgument + secondArgument + thirdArgument + fourthArgument + fifthArgument;  
        }  
      }  
    }  
  }  
}
```

3. Implemente una función que calcule el volumen de un cilindro. Incluya la versión normal y una aplicando Currying. (2 puntos)

```
var NormalCylinderVolume = (r, h) => Math.PI * Math.pow(r,2) * h;  
  
console.log("Normal: ",NormalCylinderVolume(2, 3));  
  
var CurryingCylinderVolume =(r) => {  
  return (h) => Math.PI * Math.pow(r,2) * h;  
}  
console.log("Currying: ",CurryingCylinderVolume(2)(3));
```

```
Normal: 37.69911184307752
Currying: 37.69911184307752
```

4. Cree una función joinWords que una varios parametros de tipo string. (3 puntos)

```
function joinWords(string1) {
  return (string2) => !string2 ? string1 : joinWords(`${string1} ${string2}`);
}

result = joinWords('Hello')();
console.log(result); // Hello

result = joinWords('There')('is')('no')('spoon.')();
console.log(result); // There is no spoon.
```

```
Hello
There is no spoon.
```

5. Implemente una función delayInvoc que en cada invocación incremente la variable total con el valor enviado como parametro. (3 puntos)

```
fvar total = 0;

var delayInvoc = function (a) {
  total += a;
  return function (b) {
    if(b) return delayInvoc(b);
  };
};

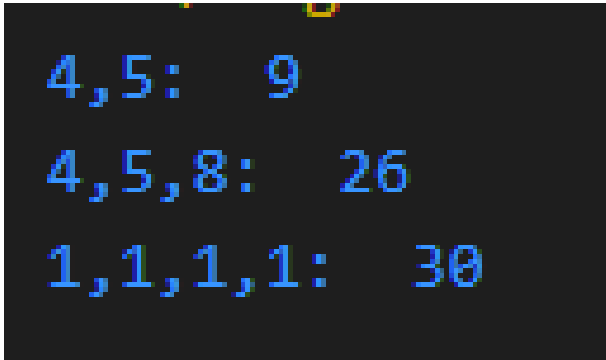
delayInvoc(4)(5);
console.log("4,5: ",total); //9
```

```

delayInvoc(4)(5)(8);
console.log("4,5,8: ",total); // 26

delayInvoc(1)(1)(1)(1);
console.log("1,1,1,1: ",total); // 30

```



6. Implemente una función curry que tome como argumento cualquier función f y retorne la versión curried de f. (4 puntos)

```

function abc(a, b, c) {
  return a+b+c;
}

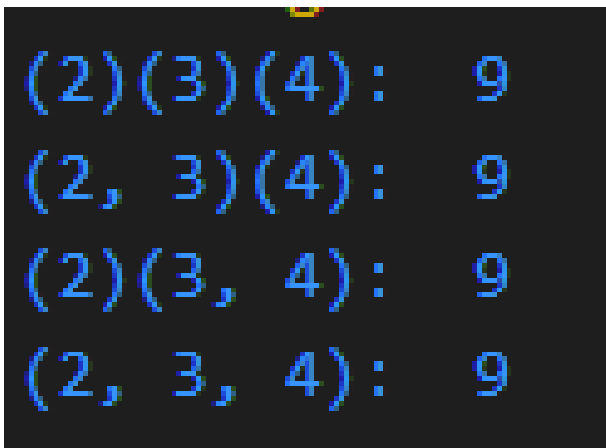
function curry(f) {
  return function curry2(...args) {
    if (args.length >= f.length) return f.apply(this, args);
    else return function curry3(...args2) {
      return curry2.apply(this, args.concat(args2));
    }
  };
}

var curriedAbc = curry(abc);

console.log("(2)(3)(4): ", curriedAbc(2)(3)(4)); // 9
console.log("(2, 3)(4): ", curriedAbc(2, 3)(4)); // 9
console.log("(2)(3, 4): ", curriedAbc(2)(3, 4)); // 9

```

```
console.log("(2, 3, 4): ", curriedAbc(2, 3, 4)) ; // 9
```



```
(2)(3)(4): 9  
(2, 3)(4): 9  
(2)(3, 4): 9  
(2, 3, 4): 9
```

GitHub: <https://github.com/KEPCU/FundamentalsOfProgrammingLanguages/tree/master/currying>