Currying

Currying is a function that *takes one argument at a time and returns a new function expecting the next argument*. It is a conversion of functions from callable as f(a,b,c)into callable as f(a)(b)(c).

Currying is a function that takes multiple arguments as input. It transform the function into a number of functions where every function will accept one argument.

```
/*Simple function*/
const add = (a, b, c)=>{
    return a+ b + c
}
console.log(add(1,2 ,3)); // 6

/* Curried Function */
const addCurry = (a) => { // takes one argument
    return (b)=>{                 //takes second argument
        return (c)=>{             //takes third argument
            return a+b+c
        }
    }
}
console.log(addCurry(1)(2)(3)); //6
```

Question-01:

evaluate(”sum”)(4)(2)⇒6

evaluate(”multiply”)(4)(2)⇒8

evaluate(”divide”)(4)(2)⇒2

evaluate(”substract”)(4)(2)⇒2

```
function sum(operation) {
    return (a) => {
        return (b) => {
        if(operation === "sum")
                  return a + b;
                    else if(operation === "multiply")
                    return a * b;
                    else if(operation === "divide")
                    return a / b;
                    else if(operation === "subtract")
                    return a - b;
                    else return "No / Invalid Operation Selected"
        }
    }
}
```

Question-02: Infinite Currying

```
function add(a) {
    return function (b) {
        if (b) return add(a + b);
        return a;
    };
}
console.log(add(5)(2)(4)(8)());
```

Question-03: Currying vs Partial Application

currying=(number of arguments == number of functions)

```
function sum(a) {
    return function (b, c) {
        return a + b + c;
    };
}
const x = sum(10);
console.log(x(5, 6));
console.log(x(3, 2));

//or
console.log(sum(20)(1, 4));
//arguments 3 but return functions 2 so ot is not currying
```