```
// Q1-
  List the side effect and convert the function to a pure function which
  does the same thing

let arr = [1, 2, 3, 4];

function f(arr) {
    for (x in arr) {
        arr[x] = 0
    }
    return arr;
}

console.log(arr);

console.log(f(arr));

console.log(arr);
```

```
// Q2 -
 Convert the following function "f" to a pure function and create an impur
e function g which is a higher order function which take result of f and p
rint it like f does it here.

let obj = {
  1: 0,
  2: 1,
  3: 2,
  4: 3,
  5: 4,
  length: 5,
};

function f() {
  for (let i = 1; i < obj.length; i++) {
    obj[i] = obj[i] + 1;
  }
  delete obj["length"];
```

```
  for (let x in obj) {
    console.log(`at index ${x} we have value ${obj[x]}`);
  }
}

f();
```

```
// Q3 -
 Write a function f that returns product of x and y with both of the follo
wing function calls

// f(x, y)
// f(x)(y)
```

```
// Q4- find the output of the following code

let a = ["a", "b"]
a[2] = a

function f(a) {
    a = a[2]
    console.log(a);
    let n = Array("a", "b")
    console.log(a[2] = n);
    console.log(a);
    console.log(n);
    a = n;
    console.log(a);
}


console.log(a);
f(a)
console.log(a);
```

```
// Options:

// 1)
// ["a", "b", ["a", "b"]]
// ["a", "b"]
// ["a", "b", ["a", "b"]]
// ["a", "b", ["a", "b"]]
// ["a", "b"]
// ["a", "b"]
// ["a", "b", ["a", "b"]]



// 2)
// [ 'a', 'b', [Circular] ]
// [ 'a', 'b', [Circular] ]
// [ 'a', 'b' ]
// [ 'a', 'b', [ 'a', 'b' ] ]
// [ 'a', 'b' ]
// [ 'a', 'b' ]
// [ 'a', 'b', [ 'a', 'b' ] ]
```

```
//Q5- create a polyfill of reduce
```

```
//Q6- How to implement setInterval of your own using setTimeout
```

```
// Q7 - find output of the following:

let count = 0;
let interval = setInterval(function () {
  console.log(count);
  count++;
}, 100);
```

```
setTimeout(function () {
  clearInterval(interval);
  interval = setInterval(function () {
    console.log(count);
    count--;
    if (count < 0) clearInterval(interval);
  });
}, 500);

// options:

// 1) Error

// 2) 0 1 2 3 4....

// 3) 0 1 2 3 4 4 3 2 1 0

// 4) 0 1 2 3 4 3 2 1 0
```

```
//Q8-
 Transducer is a higher order function which takes 3 parameter => an array
, a function used for filtering and another function to map values and ret
urns the resultant array without mutation

// which of the following definitions is/are correct?


//A
function transducer(arr, fFn, mFn) {
  let nArr = arr.filter(fFn);
  nArr = nArr.map(mFn);
  return nArr;
}


//B
```

```
function transducer(arr, fFn, mFn) {
  let nArr = [];
  for (x in arr) {
    if (fFn(arr[x])) {
      nArr.push(arr[x]);
    }
  }

  for (x in nArr) {
    nArr[x] = mFn(nArr[x]);
  }
  return nArr;
}

// Options:

// 1) A
// 2) B
// 3) Both
// 4) None
```

```
//Q9- Find the output of following :

function globalFunction(x) {
  return function outerFunction(y) {
    return function innerFunction(z) {
      return x + y + z;
    };
  };
}

let instance1 = globalFunction(2);
var instance2 = instance1(3);
console.log(instance2());
```

```
// Options:

// 1) undefined
// 2) error
// 3) NaN
// 4) 5undefined
```

```
//Q10- Find the output of the following:

let arr = ["a", "b", "c", "d", 1, 2, 3, 4];

arr.map(function (e) {
  return 2 * e;
});

(function () {
  arr.filter(function () {});
})();

console.log(arr);

let nArr;
nArr = arr.filter(function (e) {
  return Number.isInteger(e);
});
nArr = new Array();
console.log(nArr);

nArr = arr
  .filter(function (e) {
    return !Number.isInteger(e);
  })
  .map(function (e) {
    return e + 1;
  });

console.log(nArr);
```

```
// Options:

// 1)
// []
// [1, 2, 3, 4]
// ['b', 'c', 'd', 'e']

// 2)
// ["a", "b", "c", "d", 1, 2, 3, 4];
// [1, 2, 3, 4]
// ["a1","b1","c1","d1"]

// 3)
// ["a", "b", "c", "d", 1, 2, 3, 4];
// []
// ['b', 'c', 'd', 'e']

// 4)
// [ 'a', 'b', 'c', 'd', 1, 2, 3, 4 ]
// []
// [ 'a1', 'b1', 'c1', 'd1' ]
```