1. [8] Design a hospital management system.

In the system, we have doctors and patients.

Doctors and patients share fname, lname and age properties and should not duplicate those properties in their constructors.

Patients have address property and doctors have branch property..

We have 2 different types of doctors that are surgeon and physician.

Surgeons have doTheSurgery method, physicians have makeTheTreatment method.

Implement this system by using inheritance. You have to use **class keyword**. You can define other objects if necessary.

```
class Person{
      constructor(fname,lname,age){
      this.fname=fname;
    this.lname=lname;
    this.age=age
  }}

class Patient extends Person{
      constructor(fname,lname,age,address){
      super(fname,lname,age);
    this.address=address;
  }}

class Doctor extends Person{
      constructor(fname,lname,age,branch){
      super(fname,lname,age);
    this.branch=branch;
  }}
class Surgeon extends Doctor{
      constructor(fname,lname,age,branch){
      super(fname,lname,age);
    this.branch=branch;
  }
  doTheSurgery(){
  }}
class Physician extends Doctor{
      constructor(fname,lname,age,branch){
      super(fname,lname,age);
    this.branch=branch;
  }
  makeTheTreatment(){
  }
}https://jsfiddle.net/umurinan/q7bg09j2/1/
```

2. [4]

```
function Dog(name){
    this.name = name;
    this.walk = function(){
        alert(this.name + ' is walking . . ');
    }
}

let myDog = Dog('Candy');
myDog.walk();
```

Considering the code above;

Value of myDog is _____undefined_____

Output of `myDog.walk()` is _____error_____

https://jsfiddle.net/keithlevi/fnycv850/1/

3a. [5] Write an iterative function that returns true if the string is a palindrome, otherwise returns false.

3b. [5] Write a recursive function that returns true if a string is a palindrome, otherwise false.

```
function isPalindrome(str){
    if(str.length === 0) {
        return true
    }
      if(str[0] !== str[str.length-1]){
        return false
    }
  return isPalindrome(str.substring(1, str.length-1))
}
```

https://jsfiddle.net/keithlevi/nkfLdub3/2/

4. [5] Write a JavaScript function "checkExam" that returns the score of an exam.The function takes an array as it argument, which contains objects with 2 properties. One property is 'answer' holding the student answer and the other is 'key' holding the correct answer. For example

[{answer: "a", key: "a"}, {answer: "c",key: "a"}, {answer: "b", key: "b" }];→4

[{answer: "a", key: "a"}, {answer: "c",key: "a"}, {answer: "", key: "b" }];→-1

```
function checkExam(exam) {
  let score = 0;
  for (let item of exam) {
    if (item.answer === item.key) {
      score = score + 4;
    } else if (item.answer === "") {
      score = score - 1;
    } else {
      score = score - 4;
    }
  }
  return score;
}
```
https://jsfiddle.net/keithlevi/u5x43g6r/15/

5. [4] a. Write a function, absValue, that map can use to return a new array with the absolute value of each element. Below is an example usage.

```
let array =[-10, 10, 20, -20, -10, 30]
let returnVal= array.map(absValue);
console.log(returnVal);  //logs [10, 10, 20, 20, 10, 30]

function absValue(value){
          return Math.abs(value);
}
```
https://jsfiddle.net/keithlevi/9Lr5p34b/1/

b. [4] Write a second function, power, that map can use to return a new array with each element raised to the value of the exponent. Hint: This involves a closure to account for the exponent since map will only provide the number argument. Your 'power' must return a function for use by map.

```
let array = [-1, 1, 2, -2, -1, 3]
let returnVal = array.map(power(3));
console.log(returnVal); //logs [-1, 1, 8, -8, -1, 27]
function power(exponent) {
  return function(value) {
    return Math.pow(value, exponent);
  }}
```
https://jsfiddle.net/keithlevi/hdev7m5k/5/

6. [5] The following is a node structure for a linked list. Write a recursive function, doubleTheCredits, that updates the credit property of every node by doubling it. Assume the following node structure:

```
{
courseName : 'CS303',
credit : 4,
next : null
}
function doubleTheCredits(node) {//complete the function
```

```
function doubleTheCredits(node) {
  node.credit = node.credit * 2;
  if (node.next) {
    doubleTheCredits(node.next);
  }
}
let list = {  credit: 1,  next: {    credit: 2,    next: {      credit: 3,      next: null    }  }};
doubleTheCredits(list);
console.dir(list);
```
https://jsfiddle.net/keithlevi/opynbs3j/5/

7.[5] Write a recursive function, printReverse, which takes a node as a parameter. Print all the values of LinkedList in reverse order. Assume the following node structure:

```
{
      city : 'Fairfield,
      state : 'IA',
      next : null
}
function printReverse(node){//COMPLETE THIS
```

```
 function printReverse(node) {
   if (node.next) {
    printReverse(node.next)
  }
console.log(node.city);
}
```
https://jsfiddle.net/umurinan/bkqtxcvh/1/

8.[5] Write a recursive function ‘treeLog’. Its first parameter is a tree node. Its second parameter is a function that runs a test on the node and returns true or false. Recurse through the tree and log the type and value properties to the console for every node that passes the function test. Each node has the following structure. An empty children array indicates a leaf of the tree.

```
{
type: “input”,
value: “hello world”,
id: “input1”
children :[]
}

function treeLog(node, testFunc) {//COMPLETE THIS
```

```
function treeLog(node, testFunc) {
  if (testFunc(node)) {
      console.log(node.value);
  }
  let children = node.children;
  if (children) {
      for (let i = 0; i < children.length; i++) {
          treeLog(children[i],testFunc)
      }
  }
}https://jsfiddle.net/keithlevi/2cfLhprx/3/
```

9. Write a function ‘makeWithdraw’ which returns a function that keeps track of bank accounts.
   a-) makeWithdraw should return a closure that has a private variable to keep an array of bankAccount objects.
   b-) Each object in the array has properties: name, savingAccountBalance and checkingAccountBalance.
   c-) When you call the closure, ‘makeWithdraw’, with the name and withdrawal amount it will withdraw the money from the account according to the rules below:
      -If checking account has enough balance, deduct the money from checking account.
      -If checking account does not have enough balance, deduct all the money from checking account then try to deduct the rest of the money from savings account.
      -If neither checking account nor savings account has enough balance return an error.

```
function makeWithdraw() {
  let accounts = []
  return function(name, amount) {
    for (let i = 0; i < accounts.length; i++) {
      if (accounts[i].name === name) {
        if (accounts[i].checkingBalance >= amount) {
          accounts[i].checkingBalance -= amount;
        } else {
          if ((accounts[i].savingBalance +
accounts[i].checkingBalance) > amount) {
            accounts[i].checkingBalance = 0;
            amount -= accounts[i].checkingBalance;
            accounts[i].savingBalance -= amount;
          } else {
            return 'You need to earn more money !!!'
          }
        }
      }
    }
  }
}https://jsfiddle.net/umurinan/2L5pwm3t/1/
```

10a. Write a constructor function 'MakeAccount' that returns a bank account object.

1. The bank account object has three methods: withdraw, deposit, and balance.
2. The account should have a private variable, balance, which is accessed by the three methods. The private variable should not be accessible by code external to the object.
3. The balance method will return the value of the balance private variable.
4. The withdraw and deposit methods will deduct or increase the balance private variable.

```
function MakeAccount() {
  let balance = 0;
  this.withdraw = function(amt) {balance = balance - amt;},
  this.deposit = function(amt) {balance = balance + amt;},
  this.balance = function() {return balance;};
}
```
https://jsfiddle.net/keithlevi/j0gz2mhb/6/

10b. Use the constructor to create a new account object. Make a deposit of 100, then withdrawal of 50, then log the balance to the console.

```
const anAccount = new MakeAccount();
anAccount.deposit(100);
anAccount.withdraw(50);
console.log("expecting 50: " + anAccount.balance());
```

11. Repeat 10a and 10b, except this time use a JavaScript class instead of a function constructor. Instead of having a private variable holding the balance you will need to have a property for the balance. Use the _ convention to indicate that the _balance property should not be directly accessed. Use the same three method names.

```
class MakeAccount {
constructor(){
this._balance = 0;}

  withdraw(amt) {this._balance = this._balance - amt;}
  deposit(amt) {this._balance = this._balance + amt;}
  balance() {return this._balance;}}

const anAccount = new MakeAccount();
anAccount.deposit(100);
anAccount.withdraw(50);
console.log("expecting 50: " + anAccount.balance());
```
https://jsfiddle.net/keithlevi/ytvhe10d/8/

Recall the syntax for Array.reduce
arr.reduce(callback(accumulator, currentValue[, index[, array]])[, initialValue])

12a. [4] Write a callback function that work with reduce to sum all the values in an array of numbers. Your function must work for ANY array. Use the Array.reduce method to find the product of all numbers in an array.

function callback// COMPLETE THIS

```
function callback(total, current) {
    return total * current;
}
```

12b. [2] Show how you would use your callback function with reduce to find the product of [1, 2, 3, 4, 5]

```
const tot = arr1_5.reduce(callback);
```

13a. [4] Write a function, myReduce, that takes an array as its first parameter and the same callback function from the previous question as the second parameter and an initialValue as its third parameter and uses the callback function to go through the array and compute the product of the array elements.

Function myReduce(arr, cbFun, initVal) { //COMPLETE THIS

```
function myReduce(arr, callback, initial) {
    let tot = initial;
    for (let elmt of arr) {
        tot = callback(tot, elmt);
    }
    return tot;
}
```
https://jsfiddle.net/keithlevi/8dv37yjL/4/

13b. []2 Show how you would use myReduce to find the product of [1, 2, 3, 4, 5]

```
myReduce(arr1_5, callback, 1);
```

14. [7] Considering the code below, fill in the blanks.

```
let animal = {name: 'default'}

function Bird(name, age) {
  this.age = age;
  this.name = name;
}
Bird.prototype.fly = function() {
  return this.name + ' is flying';
}
peetee = new Bird("Peetee", 12);

console.log("true:  " + (peetee.age === 12));
console.log("false:  " + (peetee.__proto__.age === 12));
console.log("false:  " + (Bird.__proto__.name === "Peetee"));
console.log("false:  " + (Bird.prototype.fly() === "Peetee is
flying"));
console.log("true:  " + (peetee.fly() === "Peetee is flying"));

Bird.prototype = animal;
polly = new Bird("Polly");

console.log("false:  " + (Bird.prototype.name === "Peetee"));
console.log("false:  " + (polly.__proto__ === peetee.__proto__));
```

https://jsfiddle.net/keithlevi/j2vro1du/5/