**ES6 Part-2**

**Q1.Filter unique array members using Set.**

var items = [4,5,4,6,3,4,5,2,23,1,4,4,4]

var uniqueItems = Array.from(new Set(items));

**Q2.Find the possible combinations of a string and store them in a MAP?**

let map\_index=0;

let findCombo=function combo(map,str,l,r){

if(l==r){

map\_index++;

insertInToMap(map,map\_index,str);

}

else{

for(let i=l; i<=r; i++){

str=swap(str,l,i);

combo(map,str, l+1, r);

str = swap(str,l,i);

}

}

}

let swap= function (a, i, j)

{

let strToArr=a.split('');

temp = strToArr[i];

strToArr[i] = strToArr[j];

strToArr[j] = temp;

return strToArr.join('');

}

let insertInToMap=function (map,key,value){

map.set(key,value);

}

//Testing function

let map=new Map();

findCombo(map,"abc",0 ,2);

console.log(map);

**Q3. Write a program to implement a class having static functions**

class ClassWithStaticMethod {

static staticMethod() {

return 'static method has been called.';

}

}

console.log(ClassWithStaticMethod.staticMethod());

**Q4.Write a program to implement inheritance upto 3 classes.The Class must public variables and static functions.**

class Person{

minAge=1;

constructor(name){

this.name=name;

}

static MaxAge(){

return 125;

}

}

class Employee extends Person{

constructor(name,salary){

super(name);

this.salary=salary;

}

}

class Developer extends Employee{

constructor(name,salary,competency){

super(name,salary);

this.competency=competency;

}

}

//Testing code

let dev=new Developer('john',2000,'MEAN');

console.log(dev);

//Testing static method

console.log(Person.MaxAge());

//Testing public variables

console.log(dev.minAge);

**Q.5 Import a module containing the constants and method for calculating area of circle, rectangle, cylinder.**

**Q.6 Import a module for filtering unique elements in an array.**

**Q.7 Write a program to flatten a nested array to single level using arrow functions.**

**Q8.Implement a singly linked list in es6 and implement addFirst() addLast(), length(), getFirst(), getLast(). (without using array)**

class Node{

constructor(value){

this.data = value;

this.next = null;

}

}

class LinkedList{

constructor(value){

this.head = new Node(value);

this.tail = this.head;

// console.log(this.head);

// console.log(this.tail);

}

addFirst(value){

console.log(this.head);

let new\_node = new Node(value);

new\_node.next = this.head;

this.head = new\_node;

console.log(this.head);

}

addLast(value){

let new\_node = new Node(value);

this.tail.next = new\_node;

this.tail = new\_node;

}

length(){

let temp\_head = this.head;

let length = 0;

while(temp\_head.next != null){

length++;

temp\_head = temp\_head.next;

}

return length;

}

getFirst(){

return this.head;

}

getLast(){

return this.tail;

}

toString = () => {

let temp\_head = this.head;

let str = '';

while(temp\_head != null){

str += `${temp\_head.data}`;

if(temp\_head.next!=null){

str+=` ---> `

}

temp\_head = temp\_head.next;

}

return str;

}

}

let my\_linkedList = new LinkedList(1);

my\_linkedList.addFirst(2);

my\_linkedList.addFirst(3);

my\_linkedList.addFirst(5);

my\_linkedList.addFirst(6);

my\_linkedList.addFirst('hello, HD');

my\_linkedList.addLast("ENDDDD");

my\_linkedList.addLast("END 222222");

console.log(my\_linkedList + '');

console.log(my\_linkedList.getFirst());

console.log(my\_linkedList.getLast());

**Q.9 Implement Map and Set using Es6 ?**

//Set class implementation

class Set{

constructor(arr=[]){

if(Array.isArray(arr))

{

this.arr=arr;

}

else{

throw new Error("only array can be passed in Linked list");

}

this.arr=this.arr.filter((element,index)=>{

if (this.arr.indexOf(element) == index){

return element;

}

});

}

toString=()=>{

return this.arr.toString();

}

has(element){

return this.arr.includes(element);

}

clear(){

let b=new Stack();

return this.arr.splice(0,this.arr.length)

}

delete(element){

let findElement=this.arr.indexOf(element);

return this.arr.splice(findElement,1);

}

add(element){

if(this.arr.includes(element)){

return this.arr;

}

else{

this.arr.push(element);

return this.arr;

}

}

}

//Map class implementation

class Map{

constructor(obj={}){

this.obj=obj;pp

}

toString=()=>{

return this.obj.toString();

}

getElement(key){

return this.obj[key];

}

setElement(key,valueOfKey){

this.obj[key]=valueOfKey;

}

has(key){

return this.obj.hasOwnProperty(key)

}

}

//Testing code

let set= new Set([1,1,2,2]);

set.add(1);

set.add(2);

set.add(3);

console.log(set+'');

let map=new Map();

map.setElement(1,2);

map.setElement("a",3);

console.log(map);

**Q.10 Implementation of stack**

class Node{

constructor(data,next=null){

this.data=data;

this.next=next;

}

}

class LinkedList{

constructor(element){

this.head=new Node(element);

this.tail=this.head;

this.length=0;

}

addLast(element){

let newNode=new Node(element);

this.tail.next=newNode;

this.tail=newNode;

this.length++;

}

getLast(){

return this.tail.data;

}

addFirst(element){

let newNode =new Node(element);

// let oldHead=this.head;

newNode.next = this.head;

this.head = newNode

this.length++;

}

getFirst(element){

return this.head.data;

}

length=()=>{

return this.length;

}

toString=()=>{

let tempNode=this.head;

tempStr="";

while(tempNode !=null){

tempStr += tempNode.data+" ";

tempNode=tempNode.next;

}

return tempStr;

}

}

class Stack extends LinkedList{

constructor(element){

super(element);

}

push(element){

this.addLast(element);

}

pop(){

let lastElement=this.getLast();

let tempNode=this.head;

let prevPnt;

while(tempNode.next !=this.tail){

prevPnt=tempNode;

tempNode=tempNode.next;

}

prevPnt.next=null;

this.tail=prevPnt;

return lastElement;

}

}

//Testing code for stack

let st=new Stack(5);

st.push(5);

st.push(3);

st.push(9);

console.log(st.pop())

console.log("printing stack");

console.log(st+ "");