//InterView Question

let str="Hello, How are You";

//Covert string in only 1 array

console.log([str]);

//covert string to array with each character

console.log([...str]);

console.log(str.split(''));

//covert string to array with space or any sequence

console.log(str.split(' '));

console.log(str.split('a'));

//Replace any character

console.log(str.replace('H','w'));

console.log(str.replace(/H/g,'r'));

//Get substring from string

console.log(str.substring(6,10));

//Remove 1st character

console.log(str.substring(5,str.length));

//Remove last character

console.log(str.substring(5,str.length-1));

//Reverse string

let temp=str.split("").reverse().join("");

console.log(temp);

//Remove string before specific character

//Remove string before specific character

//Remove extra space from both side

/\*

DSA

Definition:-

Way to organize so that we can use this data efficiently.

Example:

[8,2,1,6,0,4,3,7,5]

[0,1,2,3,4,5,6,7,8]

What is algorithm ?

Steps or a process to arrange data.

Data Structure :- 2 types

Primitive Data S..:- 1.Integer ,2.Float, 3.Character, 4. Pointer

Non-Primitive Data S…- 1.Array, 2.Lists :- (a) Linear List  - i)Stack , ii)Queue

          (b)Non-Linear List- i)Graphs, ii) Trees

, 3.Floats

Operation og Data structure

1.  Traversal

2.  Insertion

3.  Deletion

4.  Searching

5.  Sorting

6.  Merging

\*/

//Traversing

// let data = [5, 8, 7, 9, 5, 45, 78, 96, 84, 75, 88, 55, 66, 4, 2, 7, 9];

// for(let i=0;i<data.length;i++){

//     document.write(`array of ${i} is ${data[i]} <br/>`);

// }

// let x=10;

// document.write(data[x]);

//------------------------------------------

// function getElement() {

//   let el = document.getElementById("element").value;

//   if (el < data.length && typeof parseInt(el)==="number") {

//     alert(data[el]);

//   } else {

//     alert("Please enter valid input");

//   }

// }

//----------------------------------------

//Insertion

// let data1 = [10, 20, 30, 50, 60];

// let newEl = 40;

// let position = 3;

// console.log(data1);

// for (let i = data1.length - 1; i >= 0; i--) {

//   if (i >= position) {

//     data1[i + 1] = data1[i];

//     if (i == position) {

//       data1[i] = newEl;

//     }

//   }

// }

// console.log(data1);

// function InsertElement() {

//   let data1 = [10, 20, 30, 50, 60];

//   let newEl = document.getElementById('newEle').value;

//   newEl=parseInt(newEl);

//   let position = document.getElementById('position').value;

//   console.log(data1);

//   for (let i = data1.length - 1; i >= 0; i--) {

//     if (i >= position) {

//       data1[i + 1] = data1[i];

//       if (i == position) {

//         data1[i] = newEl;

//       }

//     }

//   }

//   console.log(data1);

// }

//Or  -- Splice()

// let items=[10,30,40,50,60];

// items.splice(2,0,20005);

// console.log(items);

//----------------------------------------

//Delete

// let data = [10, 20, 30, 40, 50, 60];

// let position = document.getElementById("position").value;

// for (let i = position; i <= data.length - 1; i++) {

//   //   console.log(data[i]);

//   data[i] = data[i + 1];

// }

// data.length = data.length - 1;

// console.log(data);

//console.log(data.splice(2,1))

// const removeEle = () => {

//   let data = [10, 20, 30, 40, 50, 60];

//   let position = document.getElementById("position").value;

//   position = parseInt(position);

//   for (let i = position; i < data.length - 1; i++) {

//     //   console.log(data[i]);

//     data[i] = data[i + 1];

//   }

//   data.length = data.length - 1;

//   console.log(data);

// };

//--------------------------------------

//Search Element in a an array

// let data = [45, 58, 9, 5, 4, 7, 9, 20, 33, 58];

// let item = 20;

// let index = undefined;

// for (let i = 0; i <= data.length - 1; i++) {

//   if (data[i] === item) {

//     index = i;

//     break;

//   }

// }

// console.log(index);

// console.log(data.indexOf(item));//By default function using Search

// const searchElement=()=>{

//     let data = [45, 58, 9, 5, 4, 7, 9, 20, 33, 58];

// let item = document.getElementById('searchEl').value

// let index = undefined;

// for (let i = 0; i <= data.length - 1; i++) {

//   if (data[i] === parseInt(item)) {

//     index = i;

//     break;

//   }

// }

// console.log(index);

// }

//----------------------------------

//Merge Two Array  - For Loop

// let data1 = [3, 7, 12, 15, 18];

// let data2 = [10, 20, 3, 0, 4];

// let data3 = [];

// for (i = 0; i < data1.length; i++) {

//   data3[i] = data1[i];

// }

// for (let i = 0; i < data2.length; i++) {

//   data3[data1.length + 1] = data2[i];

// }

// console.log(data3);

//Merge Two Array with sorted  - While Loop

// let data1 = [3, 7, 12, 15, 18];

// let data2 = [10, 20, 3, 4];

// let data3 = [];

// let d1 = 0;

// let d2 = 0;

// let d3 = 0;

// while (d1 < data1.length && d2 < data2.length) {

//   if (data1[d1] < data2[d2]) {

//     data3[d3] = data1[d1];

//     d1++;

//     console.log(data3[d3]);

//   } else {

//     data3[d3] = data2[d2];

//     d2++;

//     console.log(data3[d3]);

//   }

//   d3++;

// }

// console.log(d1);

// while(d1<data1.length){

//     data3[d3] = data1[d1];

//     d1++;

//     d3++;

// }

// console.log(data3);

/\*

Algorithms Complexity ?

Time Complexity :- The Time complexity is mainly calculate by counting

                the number of steps to finish the execution.

Space Complexity:- Space Complexity is the amount of space required to

                    solve a problem.

    Space Complexity= Auxiliary space + Input size

Big O notation:- Use to denote complexity (both)

Asymptotic analysis

1. f(n)=5n2 + 6n + 12

2.

\*/

//Array Sorting

// let data = [58, 10, 25, 5, 69, 1, 54];

// for (let i = 0; i < data.length; i++) {

//   for (let j = 0; j < data.length; j++) {

//     if (data[j] > data[j + 1]) {

//       let temp = data[j];

//       data[j] = data[j + 1];

//       data[j + 1] = temp;

//     }

//   }

// }

// console.log(data);

//-----------------Recursion Data Structure or Direct Recursion

//Syntax

// function recurse(){

//     recurse();

// }

// recurse();

//------------------

// function recursion(x) {

//   console.log(x);

//   if (x < 10) {

//     recursion(x + 1);

//   }

// }

// let data = 0;

// recursion(data);

//-----------------------

// function fact(item){

//     if(item==0){

//         return 1;

//     }

//     return item\*fact(item-1)

// }

// let data=5;

// console.log(fact(data));

// Indirect Recursion

// let money = 100;

// let totalApple = 0;

// function buyApply(x) {

//   if (x > 0) {

//     console.log('i have',x,'rs',totalApple);

//     buyMore(x);

//   } else {

//     console.log("I don't have apply", totalApple);

//   }

// }

// function buyMore(x) {

//     totalApple++

//     buyApply(x-10)

// //   console.log("buy More", x);

// }

// buyApply(money);

//------------------------------------------------

//Debugger use

// function colors() {

//   let data = 20;

//   console.log(data);

// }

// function days() {

//   colors();

// }

// function fruits() {

//   days();

// }

// function topFunc() {

//   fruits();

// }

// topFunc();

//Head Recursion

// function test(x){

//     console.log(x)//Head Recursion

//     if(x>0){

//         test(x-1)

//     }

//     console.log(x)//Tail Recursion

// }

// let data=5

// test(data)

//Using Recursion in Reverse a array

// let data=[5,48,25,0,87,65];

// let temp;

// function customReverse(data,start,end){

//     console.log(data);

//     if(start<=end){

//         temp=data[start];

//         data[start]=data[end];

//         data[end]=temp;

//         customReverse(data,start+1,end-1)

//     }

// }

// customReverse(data,0,data.length-1)

//-----------------------STACK-------------------------------

// 1. It is a Linear Data Structure.

//2. It;s operation on happening in top.

//3. It was fixed max value.

//4. Push - Add new element

//5. Pop:- Remove element

//6. isempty:- If u check is empty or non empty.

//7. Print:- Display the elements

// Push & Pop Operation:-

// let data=[];

// let currentSize=data.length;

// let max=5;

// function push(newVal){

//     if(currentSize>=max){

//         alert('stack is full '+newVal)

//     }

//     data[currentSize]=newVal;

//     currentSize+=1;

// }

// function pop(){

//     if(currentSize>=0){

//         currentSize-=1;

//         data.length=currentSize;

//     }else{

//         alert('stack is already empty')

//     }

// }

// push(20);

// push(30);

// push(58);

// push(90);

// push(28);

// pop();

// pop();

// pop();

// pop();

// // push(10);

// console.log(data);

//-------------------------------------

//using input field & button

// let data=[];

// let currentSize=data.length;

// let max=5;

// function push(){

//     let newVal=document.getElementById('newEl').value

//     if(currentSize>=max){

//         alert('stack is full '+newVal)

//     }

//     data[currentSize]=newVal;

//     currentSize+=1;

// }

// function pop(){

//     if(currentSize>=0){

//         currentSize-=1;

//         data.length=currentSize;

//     }else{

//         alert('stack is already empty')

//     }

// }

// function display(){

//     for(let i=0;i<=currentSize;i++){

//         console.log(data[i]);

//     }

// }

// push(20);

// push(30);

// push(58);

// push(90);

// push(28);

// pop();

// pop();

// pop();

// pop();

// push(10);

// console.log(data);

//-------------------------------

//Reverse String with stack in javascript

/\*

let data=[];

let currentSize=data.length;

function push(newVal){

    data[currentSize]=newVal;

    currentSize+=1;

}

function pop(){

    laStRemovedItem=data[currentSize-1];

    currentSize-=1;

    data.length=currentSize;

    return laStRemovedItem;

}

function reverseString(item){

    for(let i=0;i<item.length;i++){

        push(item[i])

    }

    for(let i=0;i<item.length;i++){

        item[i]=pop();

    }

}

let str='saikrihna';

str=str.split("");

reverseString(str);

console.log(str.join(''));

\*/

//Queue In javascript

/\*

let queue=[];

let currentsize=queue.length;

let maxsize=5;

function enqueue(newVal) {

    if(currentsize>=maxsize){

        alert("Overflow!");

    }else{

        queue[currentsize]=newVal;

        currentsize += 1;

    }

}

function display(){

console.warn(queue);

}

function dequeue(){

    if(currentsize>0){

        for(let i=0;i<queue.length;i++){

            queue[i]=queue[i+1];

        }

        currentsize -= 1;

        queue.length=currentsize;

    }else{

        alert('queue is already fulled')

    }

}

enqueue(10)

enqueue(20)

enqueue(50)

enqueue(50)

enqueue(50)

dequeue()

dequeue()

display();

\*/

//More Operations & Input in Queue

//fRONT REAR, & isEmpty Operation

// I skip - 19,20,21,22,23 videos

//Selection Sort with JavaScript- AScending order

/\*

let item = [10, 80, 20, 52, 45, 5, 7];

function selectionSort(data) {

  let minId;

  for (let i = 0; i < data.length; i++) {

    minId = i;

    for (let j = i + 1; j < data.length; j++) {

      if (data[j] < data[minId]) {

        minId = j;

      }

    }

    let temp = data[minId];

    data[minId] = data[i];

    data[i] = temp;

  }

}

selectionSort(item);

console.log(item);

\*/

//DeSending Order in Selection Sort

//Method -1

/\*

let item = [10, 80, 20, 52, 45, 5, 7];

function selectionSort(data) {

  let minId;

  for (let i = data.length-1; i >=0; i--) {

    minId = i;

    for (let j = i - 1; j >=0; j--) {

      if (data[j] < data[minId]) {

        minId = j;

      }

    }

    let temp = data[minId];

    data[minId] = data[i];

    data[i] = temp;

  }

}

selectionSort(item);

console.log(item);

\*/

//method - 2

/\*

let item = [10, 80, 2, 45, 5, 7];

function selectionSort(data) {

  let minId;

  for (let i = 0; i < data.length; i++) {

    minId = i;

    for (let j = i + 1; j < data.length; j++) {

      if (data[j] > data[minId]) {

        minId = j;

      }

    }

    let temp = data[minId];

    data[minId] = data[i];

    data[i] = temp;

  }

}

selectionSort(item);

console.log(item);

\*/

//InSertion Sort in JavaScript

/\*

let arr = [12, 58, 47, 57, 68, 52, 1, 2, 88, 4];

function insertionsort(data) {

  let i, current, j;

  for (i = 1; i < data.length; i++) {

    current = data[i];

    j = i - 1;

    while(j>=0 && data[j]>current){

        data[j+1]=data[j];

        j--;

    }

    data[j+1]=current

  }

}

insertionsort(arr);

console.log(arr);

\*/

//Find Even Values from array, Make Double of every element and Find Array length manually

/\*

let data=[5,8,9,7,2,1,3,6,24];

let evenArray=[];

let count=0;

for(let i=0;i<data.length;i++){

    if(data[i]%2===0){

        evenArray.push(data[i]\*2);

        count++;

    }

}

// for(let i=0;i<evenArray.length;i++){

//     evenArray[i]=evenArray[i]\*2;

// }

// let count=0;

// for(let i=0;i<evenArray.length;i++){

//     count++;

// }

console.log(evenArray);

console.log(count);

\*/

//Check string are Anagram

/\* Definition

    Anagram strings are two strings built up by the same set of character

    where the order of characters is the only difference in the strings.

\*/

/\*

function checkAnagram(str1,str2){

    let objstr1={};

    if(str1.length!==str2.length){

        return false;

    }

    for(ch of str1){

        // if(!objstr1[ch]){

            objstr1[ch]=(objstr1[ch] || 0) +1

        // }

    }

    for(ch of str2){

        if(!objstr1[ch]){

            return false;

    }else{

        objstr1[ch]--;

    }

}

return true;

}

console.log(checkAnagram("hello","ehllo"));

\*/

//Check string is palindrome or not

/\*

function palindrome(data) {

  let start = 0;

  let end = data.length - 1;

  let result = true;

  while (end > start) {

    if (data[start] != data[end]) {

      result = false;

    }

    start++;

    end--;

  }

  return result;

}

let str = "level";

palindrome(str);

console.log(palindrome(str));

\*/

//Maximum Occurring character in string

/\*

let str='Saikrihna';

let strObj={};

let maxKey='';

for(let i=0;i<str.length;i++){

    // console.log(str[i]);

    let key=str[i];

    if(!strObj[key]){

        strObj[key]=0;

    };

    strObj[key]++;

    if(maxKey==''||strObj[key]>strObj[maxKey])

    {

        maxKey=key;

    }

}

console.log(maxKey);

\*/

//Object Data structure

/\*

let data={

    name:'ram',

    ageL:24,

    mail:'akchary2@Gamepad.com',

    getName:function(){

        return this.name;

    }

}

data.contact=7438877777;

console.log(data.getName());

console.log(Object.keys(data));

console.log(Object.values(data));

console.log(Object.entries(data));

for(x in data){

    console.log(x);

    console.log(data[x]);

}

\*/

//MPA DSA IN JAVASRICPT

/\*

let data=new Map([

    ['name','ram'],

    [true,'bool key'],

    [100,'hundred']

]);

data.set('color','green');

console.log(data.size);

console.log(data.has(100));

console.log(data.get(100));

console.log(data);

console.log(data.clear);

for(x of data){

    console.log(x[0]);

    console.log(x[1]);

}

data.forEach((val,key)=>{

    console.log(val);

    console.log(key);

})

\*/

//Set DSA with JAVASRICPT

/\*

let data=new Set(['ram','anil','ram'])

data.add('yam');

console.log(data.has('anil'));

console.log(data.delete('anil'));

console.log(data);

console.log(data.values());

console.log(data.keys());

console.log(data.entries());

for(x of data){

    console.log(x)

}

data.forEach((val)=>{

    console.log(val);

})

\*/

//LINKED LIST JAVASRICPT

/\*

class List {

  constructor(data) {

    this.head = {

      value: null,

      next: null,

    };

    this.tail = this.head;

    this.size=1;

  }

  appendAdd(nodeData) {

    let newNode = {

      value: nodeData,

      next: null,

    };

    this.tail.next = newNode;

    this.tail = newNode;

    this.size++;

  }

}

let list = new List(200);

list.appendAdd(300);

list.appendAdd(400);

list.appendAdd(700);

list.appendAdd(800);

list.appendAdd(200);

console.log(list);

\*/

//video skip -35 to 54