/\*

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