DSA 23 PROGRAMS FULLY WORKING WITH SOLUTIONS

1 Write a program in JavaScript to implement simple link list ?
ANS:-

```
class Node {
    constructor(data, next = null) {
                this.data = data;
                this.next = next;
             }
            }

class LinkedList {
  constructor() {
this.head = null;
 }

 insertFirst(data) {
this.head = new Node(data, this.head);
 }

  printListData() {
  let current = this.head;

 while(current) {
    console.log(current.data);
    current = current.next;
          }
         }
}

        // Usage
let ll = new LinkedList();
ll.insertFirst(100);
ll.insertFirst(200);
ll.insertFirst(300);

ll.printListData();

// Outputs: 300, 200, 100
```

2 Write a program in JavaScript to implement singly link list ?
ANS:-

```
            class Node {
    constructor(data, next = null) {
        this.data = data;
```

```
        this.next = next;
    }
}

class SinglyLinkedList {
    constructor() {
        this.head = null;
    }

    insertAtEnd(data) {
        let newNode = new Node(data);
        if(!this.head){
            this.head = newNode;
        } else {
            let current = this.head;
            while(current.next){
                current = current.next;
            }
            current.next = newNode;
        }
    }

    printListData() {
        let current = this.head;

        while(current) {
            console.log(current.data);
            current = current.next;
        }
    }
}
        // Usage
        let sll = new SinglyLinkedList();
        sll.insertAtEnd(100);
        sll.insertAtEnd(200);
        sll.insertAtEnd(300);

        sll.printListData(); //
```

Outputs: 100, 200, 300

3. Write a program in JavaScript to implement circular singly link list?

ANS:-

```
class Node {
    constructor(data, next = null) {
        this.data = data;
        this.next = next;
    }
```

```
}

class CircularSinglyLinkedList {
    constructor() {
        this.head = null;
    }

    insertAtEnd(data) {
        let newNode = new Node(data);
        if(!this.head){
            this.head = newNode;
            newNode.next = this.head;
        } else {
            let current = this.head;
            while(current.next !== this.head){
                current = current.next;
            }
            current.next = newNode;
            newNode.next = this.head;
        }
    }

    printListData() {
        let current = this.head;

        if(current !== null){
            do {
                console.log(current.data);
                current = current.next;
            } while(current !== this.head);
        }
    }
}

// Usage
let csll = new CircularSinglyLinkedList();
csll.insertAtEnd(100);
csll.insertAtEnd(200);
csll.insertAtEnd(300);

csll.printListData(); //

Outputs: 100, 200, 300
```

4. Write a program in JavaScript to implement doubly link list?

ANS:-

```
class Node {
    constructor(data, prev = null, next = null) {
        this.data = data;
        this.prev = prev;
```

```
        this.next = next;
    }
}

class DoublyLinkedList {
    constructor() {
        this.head = null;
        this.tail = null;
    }

    insertAtEnd(data) {
        let newNode = new Node(data);
        if(!this.head){
            this.head = this.tail = newNode;
        } else {
            newNode.prev = this.tail;
            this.tail.next = newNode;
            this.tail = newNode;
        }
    }

    printListData() {
        let current = this.head;

        while(current) {
            console.log(current.data);
            current = current.next;
        }
    }
}

// Usage
let dll = new DoublyLinkedList();
dll.insertAtEnd(100);
dll.insertAtEnd(200);
dll.insertAtEnd(300);

dll.printListData(); //

 Outputs: 100, 200, 300
```

5. Write a program in JavaScript to implement doubly circular link list?

ANS:-

```
class Node {
 constructor(data) {
    this.data = data;
    this.next = null;
    this.prev = null;
 }
```

```
}

class Deque {
 constructor() {
    this.head = null;
    this.tail = null;
    this.size = 0;
 }

 addFirst(data) {
    let newNode = new Node(data);
    if (this.size === 0) {
      this.head = this.tail = newNode;
      newNode.next = newNode.prev = newNode;
    } else {
      this.head.prev = newNode;
      newNode.next = this.head;
      this.head = newNode;
      this.tail.next = newNode;
      newNode.prev = this.tail;
    }
    this.size++;
 }

 addLast(data) {
    let newNode = new Node(data);
    if (this.size === 0) {
      this.head = this.tail = newNode;
      newNode.next = newNode.prev = newNode;
    } else {
      this.tail.next = newNode;
      newNode.prev = this.tail;
      this.tail = newNode;
      this.head.prev = newNode;
      newNode.next = this.head;
    }
    this.size++;
 }

 removeFirst() {
    if (this.size === 0) {
      return "Underflow";
    }
    let removedNode = this.head;
    if (this.size === 1) {
      this.head = this.tail = null;
    } else {
      this.head = this.head.next;
      this.tail.next = this.head;
      this.head.prev = this.tail;
    }
    this.size--;
    return removedNode.data;
 }
```

```
 removeLast() {
    if (this.size === 0) {
      return "Underflow";
    }
    let removedNode = this.tail;
    if (this.size === 1) {
      this.head = this.tail = null;
    } else {
      this.tail = this.tail.prev;
      this.head.prev = this.tail;
      this.tail.next = this.head;
    }
    this.size--;
    return removedNode.data;
 }

 isEmpty() {
    return this.size === 0;
 }

 size() {
    return this.size;
 }

 display() {
    let currentNode = this.head;
    let displayString = "";
    while (currentNode) {
      displayString += currentNode.data + " ";
      currentNode = currentNode.next;
      if (currentNode === this.head) {
        break;
      }
    }
    console.log(displayString);
 }
}

let deque = new Deque();
deque.addFirst(10);
deque.addFirst(20);
deque.addLast(30);
deque.addLast(40);
deque.display(); // 20 10 30 40
deque.removeFirst();
deque.removeLast();
deque.display();

output// 10 30
```

6. Write a program in JavaScript to print elements in reverse order?
ANS:-

```
class Node {
 constructor(data) {
    this.data = data;
    this.next = null;
    this.prev = null;
 }
}

class Deque {
 constructor() {
    this.head = null;
    this.tail = null;
    this.size = 0;
 }

 addFirst(data) {
    let newNode = new Node(data);
    if (this.size === 0) {
      this.head = this.tail = newNode;
      newNode.next = newNode.prev = newNode;
    } else {
      this.head.prev = newNode;
      newNode.next = this.head;
      this.head = newNode;
      this.tail.next = newNode;
      newNode.prev = this.tail;
    }
    this.size++;
 }

 removeLast() {
    if (this.size === 0) {
      return "Underflow";
    }
    let removedNode = this.tail;
    if (this.size === 1) {
      this.head = this.tail = null;
    } else {
      this.tail = this.tail.prev;
      this.head.prev = this.tail;
      this.tail.next = this.head;
    }
    this.size--;
    return removedNode.data;
 }

 displayInReverse() {
    let currentNode = this.tail;
    let displayString = "";
```

```
    while (currentNode) {
      displayString += currentNode.data + " ";
      currentNode = currentNode.prev;
      if (currentNode === this.tail) {
        break;
      }
    }
    console.log(displayString);
 }
}

let deque = new Deque();
deque.addFirst(10);
deque.addFirst(20);
deque.addLast(30);
deque.addLast(40);
deque.displayInReverse();

//output: 40 30 20 10
```

7. Write a program to copy element from one linked list to another?

ANS:-

```
        class Node {
    constructor(data, next = null) {
        this.data = data;
        this.next = next;
    }
}

class LinkedList {
    constructor() {
        this.head = null;
    }

    insertAtEnd(data) {
        let newNode = new Node(data);
        if(!this.head){
            this.head = newNode;
        } else {
            let current = this.head;
            while(current.next){
                current = current.next;
            }
            current.next = newNode;
        }
    }

    copyList() {
        let current = this.head;
```

```
        let newList = new LinkedList();
        while(current) {
            newList.insertAtEnd(current.data);
            current = current.next;
        }
        return newList;
    }

    printListData() {
        let current = this.head;

        while(current) {
            console.log(current.data);
            current = current.next;
        }
    }
}

// Usage
let ll1 = new LinkedList();
ll1.insertAtEnd(100);
ll1.insertAtEnd(200);
ll1.insertAtEnd(300);

console.log("Original List:");
ll1.printListData(); // Outputs: 100, 200, 300

let ll2 = ll1.copyList();

console.log("Copied List:");
ll2.printListData();

// Outputs: 100, 200, 300
```

8. Write a program in JavaScript to implement stack using array?

ANS:-

```
        class Stack {
 constructor() {
    this.items = [];
 }

 push(item) {
    this.items.push(item);
 }

 pop() {
    if (this.isEmpty()) {
      return "Underflow";
    }
    return this.items.pop();
```

```
 }

 peek() {
    if (this.isEmpty()) {
      return "No elements in Stack";
    }
    return this.items[this.items.length - 1];
 }

 isEmpty() {
    return this.items.length === 0;
 }

 display() {
    let displayString = "";
    for (let i = 0; i < this.items.length; i++) {
      displayString += this.items[i] + " ";
    }
    console.log(displayString);
 }
}

let stack = new Stack();
stack.push(10);
stack.push(20);
stack.push(30);
stack.push(40);
stack.display(); // 10 20 30 40
console.log(stack.pop()); // 40
console.log(stack.peek()); // 30
stack.display();
        output // 10 20 30
```

9. Write a program in JavaScript to implement stack using linked list?

ANS:-

```
        class Node {
    constructor(data, next = null) {
        this.data = data;
        this.next = next;
    }
}

class Stack {
    constructor() {
        this.top = null;
    }

    push(data) {
        this.top = new Node(data, this.top);
    }

    pop() {
```

```
        if(this.top){
            let data = this.top.data;
            this.top = this.top.next;
            return data;
        }
        return null;
    }

    peek() {
        return this.top ? this.top.data : null;
    }

    isEmpty() {
        return this.top === null;
    }
}

// Usage
let stack = new Stack();
stack.push(100);
stack.push(200);
stack.push(300);

console.log(stack.pop()); // Outputs: 300
console.log(stack.peek()); // Outputs: 200
console.log(stack.isEmpty());

// Outputs: false
```

10. Write a program in JavaScript to implement queue using array?

ANS:-

```
        class Queue {
 constructor() {
    this.items = [];
 }

 enqueue(item) {
    this.items.push(item);
 }

 dequeue() {
    if (this.isEmpty()) {
      return "Underflow";
    }
    return this.items.shift();
 }
 front() {
    if (this.isEmpty()) {
      return "No elements in Queue";
    }
```

```
    return this.items[0];
 }
 isEmpty() {
    return this.items.length === 0;
 }
 display() {
    let displayString = "";
    for (let i = 0; i < this.items.length; i++) {
      displayString += this.items[i] + " ";
    }
    console.log(displayString);
 }
}
let queue = new Queue();
queue.enqueue(10);
queue.enqueue(20);
queue.enqueue(30);
queue.enqueue(40);
queue.display(); // 10 20 30 40
console.log(queue.dequeue()); // 10
console.log(queue.front()); // 20
queue.display();

//output: 20 30 40
```

11. Write a program in JavaScript to implement queue using linked list?

ANS:-

```
class Node {
    constructor(data, next = null) {
        this.data = data;
        this.next = next;
    }
}
class Queue {
    constructor() {
        this.front = null;
        this.rear = null;
    }
    enqueue(data) {
        let newNode = new Node(data);
        if(!this.rear){
            this.front = this.rear = newNode;
        } else {
            this.rear.next = newNode;
            this.rear = newNode;
        }
    }
    dequeue() {
        if(this.front){
            let data = this.front.data;
```

```
            this.front = this.front.next;
            if(this.front === null){
                this.rear = null;
            }
            return data;
        }
        return null;
    }
    peek() {
        return this.front ? this.front.data : null;
    }
    isEmpty() {
        return this.front === null;
}
}
// Usage
let queue = new Queue();
queue.enqueue(100);
queue.enqueue(200);
queue.enqueue(300);
console.log(queue.dequeue()); // Outputs: 100
console.log(queue.peek()); // Outputs: 200
console.log(queue.isEmpty());

// Outputs: false
```

12. Write a program in JavaScript to implement merge sort using array?

ANS:-

```
function mergeSort(array) {
 if (array.length < 2) {
    return array;
 }
 const mid = Math.floor(array.length / 2);
 const left = array.slice(0, mid);
 const right = array.slice(mid);
 return merge(mergeSort(left), mergeSort(right));
}
function merge(left, right) {
 const merged = [];
 let leftIndex = 0;
 let rightIndex = 0;
 while (leftIndex < left.length && rightIndex < right.length) {
    if (left[leftIndex] < right[rightIndex]) {
      merged.push(left[leftIndex]);
      leftIndex++;
    } else {
      merged.push(right[rightIndex]);
      rightIndex++;
    }
 }
```

```
 while (leftIndex < left.length) {
    merged.push(left[leftIndex]);
    leftIndex++;
 }
 while (rightIndex < right.length) {
    merged.push(right[rightIndex]);
    rightIndex++;
 }
 return merged;
}
let array = [5, 8, 1, 4, 2, 6, 3, 7];
console.log("Original Array:");
console.log(array);
console.log("Sorted Array:");
console.log(mergeSort(array));
```

13. Write a program in JavaScript to implement Quick sort using array?

ANS:-

```
        function quickSort(arr, left = 0, right = arr.length - 1) {
    if (left < right) {
        let pivotIndex = pivot(arr, left, right);
        quickSort(arr, left, pivotIndex - 1);
        quickSort(arr, pivotIndex + 1, right);
    }
    return arr;
}

function pivot(arr, start = 0, end = arr.length - 1) {
    let pivot = arr[start];
    let swapIdx = start;

    for (let i = start + 1; i <= end; i++) {
        if (pivot > arr[i]) {
            swapIdx++;
            [arr[swapIdx], arr[i]] = [arr[i], arr[swapIdx]];
        }
    }

    [arr[start], arr[swapIdx]] = [arr[swapIdx], arr[start]];
    return swapIdx;
}

// Usage
let arr = [5, 2, 1, 8, 4, 7, 6, 3];
console.log(quickSort(arr));

// Outputs: [1, 2, 3, 4, 5, 6, 7, 8]
```

14. Write a program in JavaScript to implement circular Queue?

ANS:-

```
        class CircularQueue {
 constructor(size) {
    this.size = size;
    this.queue = new Array(size);
    this.head = this.tail = 0;
 }

 enqueue(data) {
    if (this.isFull()) {
      console.log("Queue is full.");
      return;
    }
    this.queue[this.tail] = data;
    this.tail = (this.tail + 1) % this.size;
 }

 dequeue() {
    if (this.isEmpty()) {
      console.log("Queue is empty.");
      return;
    }
    const data = this.queue[this.head];
    this.head = (this.head + 1) % this.size;
    return data;
 }

 isFull() {
    return (this.tail + 1) % this.size === this.head;
 }

 isEmpty() {
    return this.head === this.tail;
 }
}

let queue = new CircularQueue(5);
queue.enqueue(1);
queue.enqueue(2);
queue.enqueue(3);
queue.enqueue(4);
queue.enqueue(5);
console.log("Queue:");
console.log(queue.queue);

let dequeued = queue.dequeue();
console.log("Dequeued:");
console.log(dequeued);
console.log("Queue after dequeue:");
console.log(queue.queue);
```

15. Write a program in JavaScript to implement Priority Queue?

ANS:-

```
        class PriorityQueue {
    constructor() {
        this.values = [];
    }

    enqueue(val, priority) {
        this.values.push({val, priority});
        this.sort();
    };

    dequeue() {
        return this.values.shift();
    };

    sort() {
        this.values.sort((a, b) => a.priority - b.priority);
    };
}

// Usage
let pq = new PriorityQueue();
pq.enqueue("A", 3);
pq.enqueue("B", 2);
pq.enqueue("C", 1);

console.log(pq.dequeue().val);
console.log(pq.dequeue().val);
console.log(pq.dequeue().val);

 // Outputs: "C"
 // Outputs: "B"
 // Outputs: "A"
```

16. Write a program in JavaScript to implement Binary Search?

ANS:-

```
        function binarySearch(arr, low, high, x) {
    if (high >= low) {
        let mid = low + Math.floor((high - low) / 2);
        if (arr[mid] === x) return mid;
        if (arr[mid] > x) return binarySearch(arr, low, mid - 1, x);
        return binarySearch(arr, mid + 1, high, x);
    }
    return -1;
}

let arr = [2, 3, 4, 10, 40];
```

```
let x = 10;
let result = binarySearch(arr, 0, arr.length - 1, x);
(result !== -1) ? console.log("Element is present at index " + result) :
console.log("Element is not present in array");
```

17. Write a program in JavaScript to print BFS and DFS Binary Tree?

ANS:-

```
        class Node {
    constructor(value){
        this.value = value;
        this.left = null;
        this.right = null;
    }
}

class BinarySearchTree {
    constructor(){
        this.root = null;
    }

    insert(value){
        var newNode = new Node(value);
        if(this.root === null){
            this.root = newNode;
            return this;
        }
        var current = this.root;
        while(true){
            if(value === current.value) return undefined;
            if(value < current.value){
                if(current.left === null){
                    current.left = newNode;
                    return this;
                }
                current = current.left;
            } else {
                if(current.right === null){
                    current.right = newNode;
                    return this;
                }
                current = current.right;
            }
        }
    }

    BFS(){
        var node = this.root,
            data = [],
            queue = [];
```

```
        queue.push(node);

        while(queue.length){
           node = queue.shift();
           data.push(node.value);
           if(node.left) queue.push(node.left);
           if(node.right) queue.push(node.right);
        }

        return data;
    }

    DFSPreOrder(){
        var data = [];
        function traverse(node){
            data.push(node.value);
            if(node.left) traverse(node.left);
            if(node.right) traverse(node.right);
        }
        traverse(this.root);
        return data;
    }
}

// Usage
var tree = new BinarySearchTree();
tree.insert(10);
tree.insert(6);
tree.insert(15);
tree.insert(3);
tree.insert(8);
tree.insert(20);

console.log(tree.BFS());
console.log(tree.DFSPreOrder());

// Outputs: [10, 6, 15, 3, 8, 20]
// Outputs: [10, 6, 3, 8, 15, 20]
```

18. Write a program in JavaScript to print BFS and DFS Binary Tree Preorder, post Order, In order?

ANS:-

```
        // Definition for a binary tree node.
function TreeNode(val, left, right) {
 this.val = (val === undefined ? 0 : val);
 this.left = (left === undefined ? null : left);
 this.right = (right === undefined ? null : right);
}
```

```
// Helper function to create a new tree node
function createNode(val) {
 return new TreeNode(val, null, null);
}

function BFS(root) {
 if (!root) return;
 let queue = [root];
 while (queue.length) {
    let node = queue.shift();
    console.log(node.val);
    if (node.left) queue.push(node.left);
    if (node.right) queue.push(node.right);
 }
}

function DFS_preOrder(root) {
 if (!root) return;
 console.log(root.val);
 DFS_preOrder(root.left);
 DFS_preOrder(root.right);
}

function DFS_inOrder(root) {
 if (!root) return;
 DFS_inOrder(root.left);
 console.log(root.val);
 DFS_inOrder(root.right);
}

function DFS_postOrder(root) {
 if (!root) return;
 DFS_postOrder(root.left);
 DFS_postOrder(root.right);
 console.log(root.val);
}

let root = createNode(1);
root.left = createNode(2);
root.right = createNode(3);
root.left.left = createNode(4);
root.left.right = createNode(5);

console.log("BFS: ");
BFS(root);

console.log("\nDFS Preorder: ");
DFS_preOrder(root);

console.log("\nDFS Inorder: ");
DFS_inOrder(root);

console.log("\nDFS Postorder: ");
```

```
DFS_postOrder(root);
```

19. Write a program in JavaScript to implement Graph & prints its Adjacency Matrix?

ANS:-

```
        class Graph {
    constructor() {
        this.nodes = 0;
        this.adjacentList = {};
    }

    addVertex(node) {
        this.adjacentList[node] = [];
        this.nodes++;
    }

    addEdge(node1, node2) {
        this.adjacentList[node1].push(node2);
        this.adjacentList[node2].push(node1);
    }

    showConnections() {
        const allNodes = Object.keys(this.adjacentList);
        for (let node of allNodes) {
            let nodeConnections = this.adjacentList[node];
            let connections = "";
            let vertex;
            for (vertex of nodeConnections) {
                connections += vertex + " ";
            }
            console.log(node + "-->" + connections);
        }
    }
    adjacencyMatrix() {
        const allNodes = Object.keys(this.adjacentList);
        let matrix = [];
        for (let i = 0; i < this.nodes; i++) {
            matrix[i] = [];
            for (let j = 0; j < this.nodes; j++) {
                let node1 = allNodes[i];
                let node2 = allNodes[j];
                matrix[i][j] = this.adjacentList[node1].includes(node2) ? 1 : 0;
            }
        }
        return matrix;
    }
}
// Usage
let myGraph = new Graph();
myGraph.addVertex('0');
myGraph.addVertex('1');
```

```
myGraph.addVertex('2');
myGraph.addVertex('3');
myGraph.addEdge('0', '1');
myGraph.addEdge('0', '2');
myGraph.addEdge('1', '2');
myGraph.addEdge('2', '3');

myGraph.showConnections();
console.log(myGraph.adjacencyMatrix());

// Outputs: 0-->1 2 , 1-->0 2 , 2-->0 1 3 , 3-->2
// Outputs: [[0, 1, 1, 0], [1, 0, 1, 0], [1, 1, 0, 1], [0, 0, 1, 0]]
```

20. Write a program in JavaScript to implement Prism's algorithm?

ANS:-

```
        function TreeNode(val, left, right) {
    this.val = (val === undefined ? 0 : val);
    this.left = (left === undefined ? null : left);
    this.right = (right === undefined ? null : right);
}
function prismTraversal(root) {
    if (!root) return;
    let node = root;
    let prev = null;
    while (node) {
        if (prev == node.parent.right || node.left == null) {
            console.log(node.val);
            prev = node;
            node = node.right != null ? node.right : node.parent;
        } else {
            prev = node;
            node = node.left != null ? node.left : node.right;
        }
    }
}
let root = new TreeNode(1);
root.left = new TreeNode(2);
root.right = new TreeNode(3);
root.left.left = new TreeNode(4);
root.left.right = new TreeNode(5);
console.log("Prism Traversal: ");
prismTraversal(root);
```

21. Write a program in JavaScript to implement Kruskal's algorithm?

ANS:-

```
        class UnionFind {
```

```
    constructor(elements) {
        this.count = elements.length;
        this.parent = {};
        elements.forEach(e => (this.parent[e] = e));
    }
    union(a, b) {
        let rootA = this.find(a);
        let rootB = this.find(b);
        if (rootA === rootB) return;
        if (rootA < rootB) {
            if (this.parent[b] != b) this.union(this.parent[b], a);
            this.parent[b] = this.parent[a];
        } else {
            if (this.parent[a] != a) this.union(this.parent[a], b);
            this.parent[a] = this.parent[b];
        }
    }
    find(a) {
        while (this.parent[a] !== a) {
            a = this.parent[a];
        }
        return a;
    }
    connected(a, b) {
        return this.find(a) === this.find(b);
    }
}
function kruskal(nodes, edges) {
    let unionFind = new UnionFind(nodes);
    let mst = [];
    edges.sort((a, b) => a.weight - b.weight);
    for (let i = 0; i < edges.length; i++) {
        let edge = edges[i];
        if (!unionFind.connected(edge.from, edge.to)) {
            unionFind.union(edge.from, edge.to);
            mst.push(edge);
        }
    }
    return mst;
}
// Usage
let nodes = ["A", "B", "C", "D", "E"];
let edges = [
    { from: "A", to: "B", weight: 1 },
    { from: "A", to: "C", weight: 3 },
    { from: "B", to: "C", weight: 4 },
    { from: "B", to: "D", weight: 2 },
    { from: "B", to: "E", weight: 5 },
    { from: "C", to: "E", weight: 6 },
    { from: "D", to: "E", weight: 7 }
];
console.log(kruskal(nodes, edges));
```

22. Write a program in JavaScript to implement combination Sum?
ANS:-

```
        function combinationSum(candidates, target) {
    let result = [];
    let dfs = (current, currentSum, currentCombination) => {
        if (currentSum === target) {
            result.push(currentCombination.slice());
            return;
        }
        if (currentSum > target) {
            return;
        }
        for (let i = current; i < candidates.length; i++) {
            currentCombination.push(candidates[i]);
            dfs(i, currentSum + candidates[i], currentCombination);
            currentCombination.pop();
        }
    };
    dfs(0, 0, []);
    return result;
}
let candidates = [2, 3, 6, 7];
let target = 7;
console.log("Combination Sum: ");
console.log(combinationSum(candidates, target));
```

23. Write a program in JavaScript to Find GCD?
ANS:-

```
function gcd(a, b) {
    if (b === 0) {
        return a;
    }
    return gcd(b, a % b);
}

let num1 = 60;
let num2 = 48;

console.log("GCD: " + gcd(num1, num2));
```