**Job No: 08**

**Job Name: Write a program for Water Jug Problem.**

**Theory:**

The Water Jug Problem is a puzzle in AI ,It involves two jugs with different capacities, 4L and 3L, and the challenge is to figure out a series of actions using the 3L jug to end up with 2L of water in the 4L jug.

**Code:**

const jugACapacity = **parseInt**(**prompt**("Enter Jug A Capacity: "));

const jugBCapacity = **parseInt**(**prompt**("Enter Jug B Capacity: "));

let jugAWater = **parseInt**(**prompt**("Initially Water in Jug A: "));

let jugBWater = **parseInt**(**prompt**("Initially Water in Jug B: "));

const jugAFinal = **parseInt**(**prompt**("Final State of Jug A: "));

const jugBFinal = **parseInt**(**prompt**("Final State of Jug B: "));

console.**log**(`Enter Jug A Capacity: ${jugACapacity}`);

console.**log**(`Enter Jug B Capacity: ${jugBCapacity} `);

console.**log**(`Initially Water in Jug A: ${jugAWater} `);

console.**log**(`Initially Water in Jug B: ${jugBWater}`);

console.**log**(`Final State of Jug A: ${jugAFinal} `);

console.**log**(`Final State of Jug B: ${jugBFinal} `);

let count = 0;

console.**log**(

  "Operations:\n 1. Fill A\n 2. Fill B\n 3. Empty A\n 4. Empty B\n 5.

Pour A => B\n 6. Pour B => A");

while (jugAWater !== jugAFinal || jugBWater !== jugBFinal) {

  const operation = **parseInt**(**prompt**("Enter the Operation: "));

  count++;

  if (operation === 1) jugAWater = jugACapacity;

  else if (operation === 2) jugBWater = jugBCapacity;

  else if (operation === 3) jugAWater = 0;

  else if (operation === 4) jugBWater = 0;

  else if (operation === 5)

    [jugAWater, jugBWater] = [

      jugAWater - Math.**min**(jugAWater, jugBCapacity - jugBWater),

      jugBWater + Math.**min**(jugAWater, jugBCapacity - jugBWater),

    ];

  else if (operation === 6)

    [jugBWater, jugAWater] = [

      jugBWater - Math.**min**(jugBWater, jugACapacity - jugAWater),

      jugAWater + Math.**min**(jugBWater, jugACapacity - jugAWater),

    ];

  else console.**log**("Invalid operation. Please enter a valid operation number.");

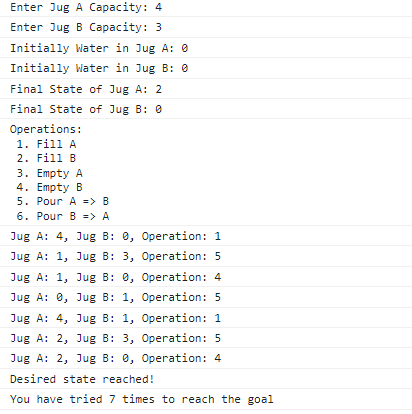
  console.**log**(`Jug A: ${jugAWater}, Jug B: ${jugBWater}, Operation: ${operation}`);

}

console.**log**("Desired state reached!");

console.**log**(`You have tried ${count} times to reach the goal`);

**Input/Output:**



**Job No: 09**

**Job Name: Write a program for Tower Of Hanoi Of Hanoi Problem.**

**Theory:** The Tower of Hanoi problem is a classic puzzle. The objective is to transfer a stack of disks from one peg to another, abiding by specific rules including moving only one disk at a time and avoiding placing bigger disks over smaller ones.

**Code:**

const towers = [[], [], []];

function **initializeTowers**(numDisks) {

  for (let i = numDisks; i > 0; i--) {

    towers[0].**push**(i);

  }

}

function **displayTowers**(source, target) {

  console.**log**(`${source === undefined? "": "Move: " + (source + 1) + " to " + (target + 1)}`);

  for (let i = 0; i < 3; i++) {

    console.**log**(`Tower ${i + 1}: ${towers[i].**join**(", ")}`);

  }

  console.**log**();

}

function **moveDisk**(sourceTower, targetTower) {

  const disk = towers[sourceTower].**pop**();

  towers[targetTower].**push**(disk);

**displayTowers**(sourceTower, targetTower);

**checkWin**();

}

function **checkWin**() {

  if (towers[0].length === 0 && towers[1].length === 0) {

    console.**log**("Congratulations! You've solved the Tower of Hanoi!");

  }

}

function **promptMove**() {

  const input = **prompt**("Enter source tower and target tower (e.g., 1 3):");

  const [source, target] = input.**split**(" ").**map**(**Number**);

  if (source >= 1 && source <= 3 && target >= 1 && target <= 3 &&

    source !== target) {

   if (towers[target - 1].length === 0 ||

      towers[source - 1][towers[source - 1].length - 1] <

       towers[target - 1][towers[target - 1].length - 1]) {

**moveDisk**(source - 1, target - 1);

**promptMove**();

    }

}

const numDisks = 3;

**initializeTowers**(numDisks);

**displayTowers**();

**promptMove**();

**Input/Output:**

