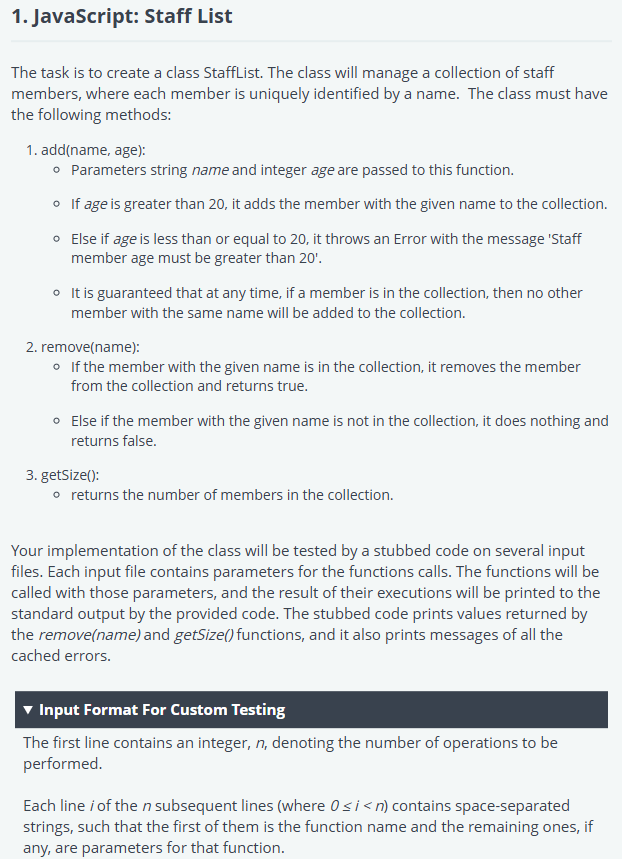
# JavaScript: Staff List



'use strict';

const fs = require('fs');

process.stdin.resume();

process.stdin.setEncoding('utf-8');

let inputString = '';

let currentLine = 0;

process.stdin.on('data', function(inputStdin) {

  inputString += inputStdin;

});

process.stdin.on('end', function() {

  inputString = inputString.split('\n');

  main();

});

function readLine() {

  return inputString[currentLine++];

}

class StaffList {

    constructor() {

        this.staffMembers = [];

    }

    add(name, age) {

        if (age > 20) {

            this.staffMembers.push({ name, age });

        } else {

            throw new Error("Error: Staff member age must be greater than 20");

        }

    }

    remove(name) {

        const index = this.staffMembers.findIndex(member => member.name === name);

        if (index !== -1) {

            this.staffMembers.splice(index, 1);

            return true;

        }

        return false;

    }

    getSize() {

        return this.staffMembers.length;

    }

}

function main() {

    const ws = fs.createWriteStream(process.env.OUTPUT\_PATH);

    const obj = new StaffList();

    const operationCount = parseInt(readLine().trim());

    for (let i = 1; i <= operationCount; i++) {

        const operationInfo = readLine().trim().split(' ');

        try {

            let res;

            if (operationInfo[0] === 'add') {

                obj.add(operationInfo[1], parseInt(operationInfo[2]));

            } else if (operationInfo[0] === 'remove') {

                res = obj.remove(operationInfo[1]);

                ws.write(`${res}\n`);

            } else if (operationInfo[0] === 'getSize') {

                res = obj.getSize();

                ws.write(`${res}\n`);

            }

        } catch (e) {

            ws.write(`${e.message}\n`);  // Changed from `e` to `e.message` to match the expected output

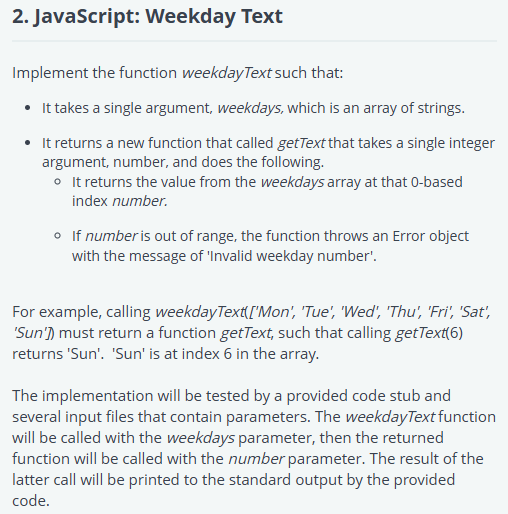
        }

    }

    ws.end();

}

# JavaScript: Weekday Text



'use strict';

const fs = require('fs');

process.stdin.resume();

process.stdin.setEncoding('utf-8');

let inputString = '';

let currentLine = 0;

process.stdin.on('data', function(inputStdin) {

    inputString += inputStdin;

});

process.stdin.on('end', function() {

    inputString = inputString.split('\n');

    main();

});

function readLine() {

    return inputString[currentLine++];

}

function weekdayText(weekdays) {

    return function getText(number) {

        if (number < 0 || number >= weekdays.length) {

            throw new Error('Invalid weekday number');

        }

        return weekdays[number];

    };

}

function main() {

    const ws = fs.createWriteStream(process.env.OUTPUT\_PATH);

    const weekdays = readLine().trim().split(' ');

    const getText = weekdayText(weekdays);

    try {

        const value = getText(parseInt(readLine().trim()));

        ws.write(value);

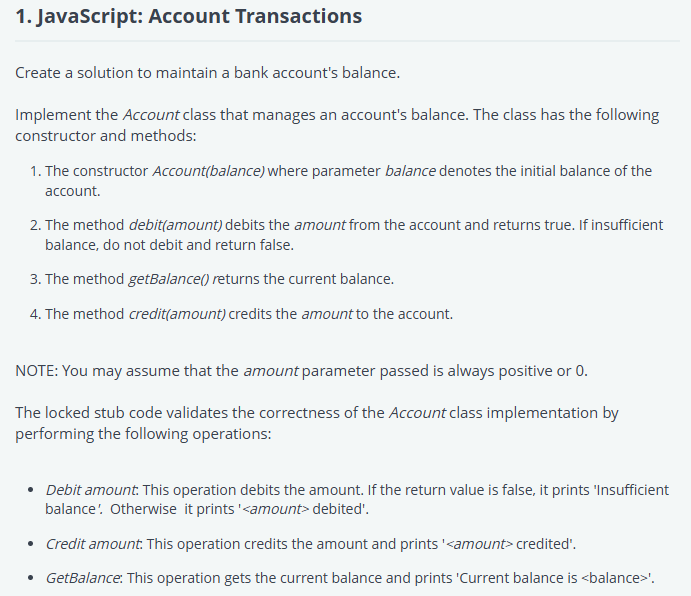
    } catch(e) {

        ws.write(`${e}`);

    }

}

# JavaScript: Account Transactions



'use strict';

const fs = require('fs');

process.stdin.resume();

process.stdin.setEncoding("ascii");

let inputString = "";

let currentLine = 0;

process.stdin.on("data", function (chunk) {

    inputString += chunk;

});

process.stdin.on("end", function () {

    inputString = inputString.split('\n');

    main();

});

function readLine() {

  return inputString[currentLine++];

}

class Account {

    constructor(balance) {

        this.balance = balance;  // Initialize with the starting balance

    }

    // Method to debit the amount from the account

    debit(amount) {

        if (this.balance < amount) {

            return false;  // Insufficient balance

        } else {

            this.balance -= amount;  // Deduct the amount from the balance

            return true;

        }

    }

    // Method to get the current balance

    getBalance() {

        return this.balance;

    }

    // Method to credit the amount to the account

    credit(amount) {

        this.balance += amount;  // Add the amount to the balance

    }

}

function main() {

    const ws = fs.createWriteStream(process.env.OUTPUT\_PATH);

    const initialBalance = parseInt(readLine().trim());

    const accountObj = new Account(initialBalance);

    ws.write(`Account created with initial balance of ${initialBalance}\n`);

    let numberOfOperations = parseInt(readLine().trim());

    while (numberOfOperations-- > 0) {

        const inputs = readLine().trim().split(' ');

        const operation = inputs[0];

        const amount = parseInt(inputs[1]);

        switch(operation) {

            case 'Debit':

                if (accountObj.debit(amount)) {

                    ws.write(`${amount} debited\n`);

                } else {

                    ws.write(`Insufficient balance\n`);

                }

                break;

            case 'Credit':

                accountObj.credit(amount);

                ws.write(`${amount} credited\n`);

                break;

            case 'GetBalance':

                const currentBalance = accountObj.getBalance();

                ws.write(`Current balance is ${currentBalance}\n`);

            default:

                break;

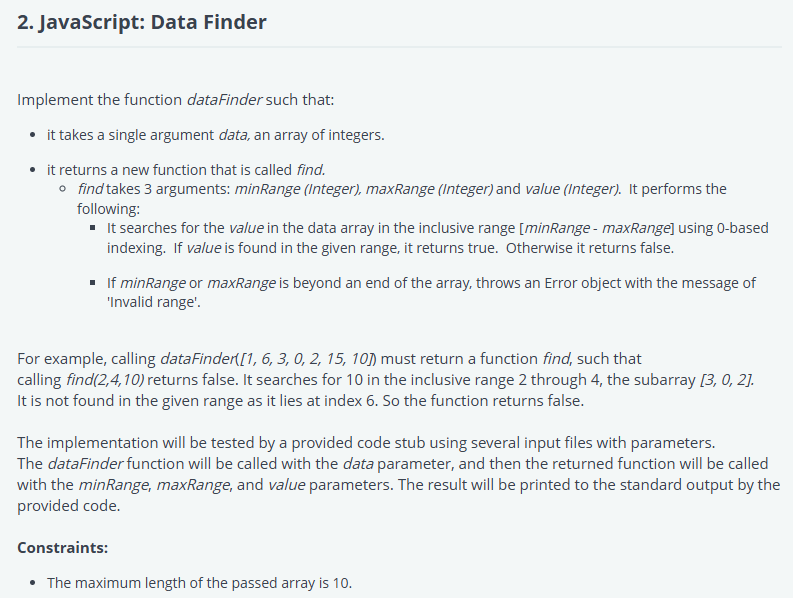
        }

    }

    ws.end();

}

# JavaScript: Data Finder



'use strict';

const fs = require('fs');

process.stdin.resume();

process.stdin.setEncoding("ascii");

let inputString = "";

let currentLine = 0;

process.stdin.on("data", function (chunk) {

    inputString += chunk;

});

process.stdin.on("end", function () {

    inputString = inputString.split('\n');

    main();

});

function readLine() {

  return inputString[currentLine++];

}

// Function to find the value in the given range

function dataFinder(data) {

    return function find(minRange, maxRange, value) {

        // Check if minRange and maxRange are valid

        if (minRange < 0 || maxRange >= data.length) {

            throw new Error("Invalid range");

        }

        // Search for the value in the range [minRange, maxRange]

        for (let i = minRange; i <= maxRange; i++) {

            if (data[i] === value) {

                return true;  // Value found

            }

        }

        return false;  // Value not found

    };

}

function main() {

    const ws = fs.createWriteStream(process.env.OUTPUT\_PATH);

    const data = readLine().trim().split(' ');

    const finalData = data.map(val => parseInt(val));

    const join = dataFinder(finalData);

    try {

        const inputs = readLine().trim().split(' ');

        const result = join(parseInt(inputs[0]), parseInt(inputs[1]), parseInt(inputs[2]));

        ws.write(result.toString());

    } catch(e) {

        ws.write(`${e}`);

    }

    ws.end();

}