function plusMinus(arr) {

    const n = arr.length;

    let pos = 0, neg = 0, zero = 0;

    for (let i = 0; i < n; i++) {

        if(arr[i] > 0) pos++;

    else if(arr[i] < 0) neg++;

    else zero++;

}

console.log((pos / n).toFixed(6));

console.log((neg / n).toFixed(6));

console.log((zero / n).toFixed(6));

}

function minMax(arr) {

    arr.sort((a, b) => a - b);

    let minSum = arr.slice(0, 4).reduce((a, b) => a + b, 0);

    let maxSum = arr.slice(1, 5).reduce((a, b) => a + b, 0);

    console.log(minSum + " " + maxSum);

}

function timeConversion(s) {

    let hour = parseInt(s.slice(0, 2)); // Extract the hour part

    const rest = s.slice(2, 8); // Extract the rest of the string (minutes, seconds, and AM/PM)

    const ampm = s.slice(8,10); // Extract AM/PM

    if(ampm ==="AM") {

        if(hour === 12) hour = 0; // Convert 12 AM to 00

    } else { // PM case

        if(hour !== 12) hour += 12; // Convert PM hour to 24-hour format

        }

        // Format the hour to always be two digits

        const hourStr = hour.toString().padStart(2, '0');

        return hourStr + rest; // Return the formatted time

    }

function findMedian(arr) {

    // Sort the array in ascending order

    arr.sort((a, b) => a - b);

    // Find the middle index

    const mid = Math.floor(arr.length / 2);

    // Since the problem usually expects an odd-length array,

    // the median is the middle element after sorting

    return arr[mid];

}

function lonelyInteger(a) {

    return a.reduce((acc, num) => acc ^ num, 0); // XOR all elements

// The result will be the lonely integer

}

function diagonalDifference(arr) {

    let primarySum = 0;

    let secondarySum = 0;

    const n = arr.length;

    for(let i = 0; i < n; i++) {

        primarySum += arr[i][i]; // Sum of primary diagonal

        secondarySum += arr[i][n - 1 - i]; // Sum of secondary diagonal

    }

    return Math.abs(primarySum - secondarySum); // Return the absolute difference

}

function countingSort(arr) {

    let freq = Array(100).fill(0); // Initialize frequency array of 100 zeros

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

        freq[arr[i]]++; // Increment the frequency of the element

    }

    return freq;

}

function flippingMatrix(matrix) {

    const n = matrix.length / 2; // Size of the submatrix

    let sum = 0;

    for(let i = 0; i < n; i++) {

        for(let j = 0; j < n; j++) {

            // Calculate the maximum value from the four possible positions

            sum += Math.max(

                matrix[i][j],

                matrix[i][2 \* n - 1 - j],

                matrix[2 \* n - 1 - i][j],

                matrix[2 \* n - 1 - i][2 \* n - 1 - j]

            );

        }

    }

    return sum;

}

function processData(inputString) {

    console.log("1 st row of code");

    console.log(inputString);

}

function towerBreaker(n, m) {  // If all towers have height 1, no moves can be made, so Player 2 wins.

    // If the number of towers is even, Player 2 can always mirror Player 1's moves and win.

    // Otherwise, Player 1 can win.

    if(m === 1 || n % 2 === 0) {

        return 2; // Player 2 wins if m is 1 or n is even

    } else {

        return 1; // Player 1 wins if m > 1 and n is odd

    }

}

function caesarCipher(s, k) {

    // Normalize k to avoid unnecessary rotations

    k = k % 26;

    let result = '';

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

        let char = s[i];

        if (char >= 'a' && char <= 'z') {

            // Lowercase letter

            let code = ((char.charCodeAt(0) - 97 + k) % 26) + 97;

            result += String.fromCharCode(code);

        } else if (char >= 'A' && char <= 'Z') {

            // Uppercase letter

            let code = ((char.charCodeAt(0) - 65 + k) % 26) + 65;

            result += String.fromCharCode(code);

        } else {

            // Non-alphabetic character

            result += char;

        }

    }

    return result;

}

function palindromeIndex(s) {

    let left = 0;

    let right = s.length - 1;

    while (left < right) {

        if (s[left] !== s[right]) {

            // Check if skipping the left character makes a palindrome

            if (isPalindrome(s, left + 1, right)) {

                return left;

            }

            // Check if skipping the right character makes a palindrome

            if (isPalindrome(s, left, right - 1)) {

                return right;

            }

            // No single removal can make it a palindrome

            return -1;

        }

        left++;

        right--;

    }

    // The string is already a palindrome

    return -1;

}

function isPalindrome(s, i, j) {

    while (i < j) {

        if (s[i] !== s[j]) return false;

        i++;

        j--;

    }

    return true;

}

function main() {

    var i = 4;

    var d = 4.0;

    var s = "HackerRank";

    // Declare second integer, double, and String variables.

    var i2;

    var d2;

    var s2;

    // Read and save an integer, double, and String to your variables.

    i2 = parseInt(readLine());

    d2 = parseFloat(readLine());

    s2 = readLine();

    // Print the sum of both integer variables on a new line.

    console.log(i + i2);

    // Print the sum of the double variables on a new line.

    console.log((d + d2).toFixed(1));

// Concatenate and print the String variables on a new line

    console.log(s + s2);

    // The 's' variable above should be printed first.

}

function flippingMatrix(matrix) {

    const n = matrix.length / 2; // Size of the submatrix

    let sum = 0;

    for(let i = 0; i < n; i++) {

        for(let j = 0; j < n; j++) {

            // Calculate the maximum value from the four possible positions

            sum += Math.max(

                matrix[i][j],

                matrix[i][2 \* n - 1 - j],

                matrix[2 \* n - 1 - i][j],

                matrix[2 \* n - 1 - i][2 \* n - 1 - j]

            );

        }

    }

    return sum;

}

function findMissingNumber(arr) {

    const n = arr.length + 1; // since one number is missing

    const expectedSum = (n \* (n + 1)) / 2;

    const actualSum = arr.reduce((sum, num) => sum + num, 0);

    return expectedSum - actualSum;

}

function fibonacci(n) {

    if (n <= 1) return n;

    return fibonacci(n - 1) + fibonacci(n - 2);

  }

// Merge two sorted linked lists

console.log(`\*\*\*\*\*\*\*\*\*\*\*Merge two sorted linked lists\*\*\*\*\*\*\*\*\*\*\*\*\*\*`);

// Definition for singly-linked list.

class ListNode {

    constructor(val, next = null) {

        this.val = val;

        this.next = next;

}

}

// Function to merge two sorted linked lists

function mergeTwoLists(l1, l2) {

let dummy = new ListNode(0);

let current = dummy;

while (l1 !== null && l2 !== null) {

    if(l1.val < l2.val) {

        current.next = l1;

        l1 = l1.next;

    } else {

        current.next = l2;

        l2 = l2.next;

    }

    current = current.next;

}

// Attach the remaining nodes, if any

current.next = l1 !== null ? l1 : l2;

return dummy.next;

}

// Helper function to create a linked list from an array

function arrayToList(arr) {

    let dummy = new ListNode(0);

    let current = dummy;

    for ( let val of arr) {

        current.next = new ListNode(val);

        current = current.next;

    }

    return dummy.next;

}

// Helper function to print linked list as array

function listToArray(head) {

    let arr = [];

    while (head) {

        arr.push(head.val);

        head = head.next;

    }

    return arr;

}

function isBalanacedParentheses(str) {

    let stack = [];

    for (let char of str) {

        if ( char === '(') {

            stack.push(char);

        } else if(char === ')') {

            if(stack.length === 0) {

                return false;

            }

            stack.pop();

        }

    }

    return stack.length === 0;

}

// Running total calculator

function runningTotal(arr, field) {

    let total = 0;

    return arr.map(item => {

      total += Number(item[field] || 0);

      return { ...item, [`${field}\_runningTotal`]: total };

    });

  }

  // Moving average calculator

  function movingAverage(arr, field, windowSize) {

    return arr.map((item, idx, src) => {

      const window = src.slice(Math.max(0, idx - windowSize + 1), idx + 1);

      const avg = window.reduce((sum, row) => sum + Number(row[field] || 0), 0) / window.length;

      return { ...item, [`${field}\_movingAvg`]: avg };

    });

  }

  function groupBy(arr, keyFn) {

    return arr.reduce((acc, item) => {

      const key = keyFn(item);

      if (!acc[key]) acc[key] = [];

      acc[key].push(item);

      return acc;

    }, {});

  }

  function removeDuplicates(arr, keyFn) {

    const seen = new Set();

    return arr.filter(item => {

      const key = keyFn(item);

      if (seen.has(key)) return false;

      seen.add(key);

      return true;

    });

  }

  // Simple CSV parser

function parseCSV(text) {

    const [headerLine, ...lines] = text.trim().split('\n');

    const headers = headerLine.split(',').map(h => h.trim());

    return lines.map(line => {

      const values = line.split(',').map(v => v.trim());

      return Object.fromEntries(headers.map((h, i) => [h, values[i]]));

    });

  }