Department of Computer Science

University of Engineering and Technology, Lahore

Write the C++ Functions of the following problems.

Problem 1:

Write a C++ function that is given a string as input, it returns true if its length is even and false if the length is odd.

Test Cases

Input	Output	Explanation
Enter a String: apples	true	// The word "apples" has 6 characters. // 6 is an even number, so the program outputs true.
Enter a String: pears	false	// "pears" has 5 letters, and 5 is odd. // Therefore the program outputs false.
Enter a String: cherry	false	

Note: Make all the arrays and their indexes global (only the arrays and the variable containing their indexes) so that you can use those in any function.

Problem 2:

Create a main that takes an array of numbers as input from the user and then calls a function sevenBoom() that returns "Boom!" if the digit 7 appears in the array. Otherwise, return "there is no 7 in the array".

Test Cases:

- [1, 2, 3, 4, 5, 6, 7]
  sevenBoom() ⟶ "Boom!"
  // 7 contains the number seven.

- [8, 6, 33, 100]
  sevenBoom() ⟶ "there is no 7 in the array"
  // None of the items contain 7 within them.

- [2, 55, 60, 97, 86]
  sevenBoom() ⟶ "Boom!"
  // 97 contains the number seven.

Problem 3:
Create a function that checks in an array (slot machine outcome) and returns true if all elements in the array are identical, and false otherwise. The array will contain 4 elements.

Notes

- The elements must be exactly identical for there to be a jackpot.

Test Cases:
- ["@", "@", "@", "@"]
  testJackpot() ⟶ true

- ["abc", "abc", "abc", "abc"]
  testJackpot() ⟶ true

- ["SS", "SS", "SS", "SS"]
  testJackpot() ⟶ true

- ["&&", "&", "&&&", "&&&&"]
  testJackpot() ⟶ false

- ["SS", "SS", "SS", "Ss"]
  testJackpot() ⟶ false

Problem 4:
Create a C++ function that performs an even-odd transform to an array, n times. One even-odd transformation is

a. Adds two (+2) to each odd integer.
b. Subtracts two (-2) to each even integer.

Test Cases

Input	Output	Explanation
Enter the array: [3, 4, 9] Enter number of times even-odd transformation need to be done: 3	[9, -2, 15]	// Since even-odd transformation needs to be applied 3 times [3, 4, 9] becomes => [5, 2, 11] => [7, 0, 13] => [9, -2, 15]
Enter the array: [0, 0, 0] Enter number of times even-odd transformation need to be done: 10	[-20, -20, -20]	
Enter the array: [1, 2, 3] Enter number of times even-odd transformation need to be done: 1	[3, 0, 5]	

Problem 5:

Make a C++ Function that is Given two strings, find the number of common characters between them and then return that count.

Example:

For s1 = "aabcc" and s2 = "adcaa", the output should be

solution(s1, s2) = 3.

Strings have 3 common characters; 2 "a"s and 1 "c".

Problem 6:

When coloring a striped pattern, you may start by coloring each square *sequentially*, meaning you spend time needing to *switch coloring pencils*.

Create a function where given an *array of colors* cols, return how long it takes to color the whole pattern. Note the following times:

- It takes **1 second** to *switch between pencils*.
- It takes **2 seconds** to *color a square*.

See the example below for clarification.

Examples:

- ["Red", "Blue", "Red", "Blue", "Red"]
  color_pattern_times() ⟶ 14
  // There are 5 colors so it takes 2 seconds to color each one (2 x 5 = 10).
  // You need to switch the pencils 4 times and it takes 1 second to switch (1 x 4 = 4).
  // 10 + 4 = 14

- ["Blue"]
  colorPatternTimes() ⟶ 2

- ["Red", "Yellow", "Green", "Blue"]
  colorPatternTimes() ⟶ 11

- ["Blue", "Blue", "Blue", "Red", "Red", "Red"]
  colorPatternTimes() ⟶ 13

Note:

- Only change coloring pencils if the next color is different to the color before it.
- Return a number in *seconds*.

Problem 7:

Your local bank has decided to upgrade its ATM machines by incorporating motion sensor technology. The machines now interpret a series of consecutive dance moves in place of a PIN number.

Create a function that converts a customer's PIN number to its dance equivalent. There is one dance move per digit in the PIN number. A list of dance moves is given below.

MOVES = ["Shimmy", "Shake", "Pirouette", "Slide", "Box Step", "Headspin", "Dosado", "Pop", "Lock", "Arabesque"];

Notes:

- Each dance move will be selected from a list by index based on the current digit's value plus that digit's index value. If this value is greater than the last index value of the dance list, it should cycle to the beginning of the list.
- Valid input will always be a string of four digits. Output will be on the console.
- If the input is not four valid numbers, output the string, "Invalid input."

Test Cases:

- danceConvert("0000") → "Shimmy", "Shake", "Pirouette", "Slide"
- danceConvert("3856") → "Slide", "Arabesque", "Pop", "Arabesque"
- danceConvert("9999") → "Arabesque", "Shimmy", "Shake", "Pirouette"
- danceConvert("32a1") → "Invalid input."