Task:

**Test a function that checks whether a string is a palindrome. Use Google Test’s parameterized testing features to test multiple cases efficiently.**

**TEST.CPP:**

**#include <gtest/gtest.h>**

**#include "palindrome.cpp"**

**struct PalindromeTest : public ::testing::TestWithParam<std::pair<std::string, bool>> {};**

**TEST\_P(PalindromeTest, CheckIfPalindrome) {**

**auto param = GetParam();**

**EXPECT\_EQ(isPalindrome(param.first), param.second);}**

**INSTANTIATE\_TEST\_SUITE\_P(**

**PalindromeTests,**

**PalindromeTest,**

**::testing::Values(**

**std::make\_pair("madam", true),**

**std::make\_pair("racecar", true),**

**std::make\_pair("apple", false),**

**std::make\_pair("level", true),**

**std::make\_pair("hello", false),**

**std::make\_pair("noon", true),**

**std::make\_pair("Abba", false) ));**

**TEXT.CPP:**

**#include "palindrome.h"**

**bool isPalindrome(const std::string& str) {**

**int left = 0, right = str.length() - 1;**

**while (left < right) {**

**if (str[left] != str[right])**

**return false;**

**left++;**

**right--;**

**}**

**return true;**

**}**

**Palindrome.h:**

**#pragma once**

**#include <string>**

**bool isPalindrome(const std::string& str);**

**Pal.cpp:**

**#include <iostream>**

**using namespace std;**

**#include "palindrome..h"**

**int main(){**

**cout << "Is 'madam' a palindrome? " << isPalindrome("madam") << endl;**

**cout << "Is 'hello' a palindrome? " << isPalindrome("hello") << endl;**

