***Q1. Write a program in a function named q1, to implement a recursive function to calculate the product of two numbers without using the multiplication operator.***

|  |
| --- |
| #include <iostream>  // Recursive function to calculate the product of two numbers  int multiply(int a, int b) {  // Base case: if one of the numbers is 0, the product is 0  if (a == 0 || b == 0) {  return 0;  }  // Recursive case: add 'a' to the product of 'a' and 'b-1'  return a + multiply(a, b - 1);  }  // Function to implement the recursive product calculation  int q1(int num1, int num2) {  // Handle the case when one of the numbers is negative  if (num1 < 0 && num2 < 0) {  // If both numbers are negative, convert them to positive  num1 = -num1;  num2 = -num2;  } else if (num1 < 0) {  // If only the first number is negative, negate the result  return -multiply(-num1, num2);  } else if (num2 < 0) {  // If only the second number is negative, negate the result  return -multiply(num1, -num2);  }  // For positive numbers, call the multiply function  return multiply(num1, num2);  }  int main() {  // Example usage  int num1, num2;  std::cout << "Enter the first number: ";  std::cin >> num1;  std::cout << "Enter the second number: ";  std::cin >> num2;  // Calculate and display the product using the q1 function  int result = q1(num1, num2);  std::cout << "Product: " << result << std::endl;  return 0;  } |

***Q2. Write a program in a function named q2, to implement a recursive function to calculate the sum of digits of a given number.***

|  |
| --- |
| ***#include <iostream>***  ***// Recursive function to calculate the product of two numbers***  ***int multiply(int a, int b) {***  ***// Base case: if one of the numbers is 0, the product is 0***  ***if (a == 0 || b == 0) {***  ***return 0;***  ***}***  ***// Recursive case: add 'a' to the product of 'a' and 'b-1'***  ***return a + multiply(a, b - 1);***  ***}***  ***// Function to implement the recursive product calculation***  ***int q1(int num1, int num2) {***  ***// Handle the case when one of the numbers is negative***  ***if (num1 < 0 && num2 < 0) {***  ***// If both numbers are negative, convert them to positive***  ***num1 = -num1;***  ***num2 = -num2;***  ***} else if (num1 < 0) {***  ***// If only the first number is negative, negate the result***  ***return -multiply(-num1, num2);***  ***} else if (num2 < 0) {***  ***// If only the second number is negative, negate the result***  ***return -multiply(num1, -num2);***  ***}***  ***// For positive numbers, call the multiply function***  ***return multiply(num1, num2);***  ***}***  ***int main() {***  ***// Example usage***  ***int num1, num2;***  ***std::cout << "Enter the first number: ";***  ***std::cin >> num1;***  ***std::cout << "Enter the second number: ";***  ***std::cin >> num2;***  ***// Calculate and display the product using the q1 function***  ***int result = q1(num1, num2);***  ***std::cout << "Product: " << result << std::endl;***  ***return 0;***  ***}*** |

***Q3. Write a program in a function named q3, to implement a recursive function to determine if a given number is a prime or not.***

|  |
| --- |
| ***#include <iostream>***  ***#include <cmath>***  ***// Recursive function to check if a number is prime***  ***bool isPrimeRecursive(int num, int divisor) {***  ***// Base case: if the divisor reaches the square root of the number, it's prime***  ***if (divisor > sqrt(num)) {***  ***return true;***  ***}***  ***// Base case: if the number is divisible by the current divisor, it's not prime***  ***if (num % divisor == 0) {***  ***return false;***  ***}***  ***// Recursive case: check with the next divisor***  ***return isPrimeRecursive(num, divisor + 1);***  ***}***  ***// Function to implement the recursive prime check***  ***bool q3(int num) {***  ***// Handle the case when the number is less than 2***  ***if (num < 2) {***  ***return false;***  ***}***  ***// Call the isPrimeRecursive function starting with divisor = 2***  ***return isPrimeRecursive(num, 2);***  ***}***  ***int main() {***  ***// Example usage***  ***int number;***  ***std::cout << "Enter a number: ";***  ***std::cin >> number;***  ***// Check and display if the number is prime using the q3 function***  ***if (q3(number)) {***  ***std::cout << number << " is a prime number." << std::endl;***  ***} else {***  ***std::cout << number << " is not a prime number." << std::endl;***  ***}***  ***return 0;***  ***}*** |

***Q4.*** ***Write a program in a function named q4, to implement a recursive function to calculate the sum of even and odd numbers in a given range.***

|  |
| --- |
| #include <iostream>  // Recursive function to calculate the sum of even and odd numbers in a range  void sumEvenOddRecursive(int start, int end, int &sumEven, int &sumOdd) {  // Base case: if start is greater than end, stop recursion  if (start > end) {  return;  }  // Check if the current number (start) is even or odd and update the sums  if (start % 2 == 0) {  sumEven += start;  } else {  sumOdd += start;  }  // Recursive case: move to the next number in the range  sumEvenOddRecursive(start + 1, end, sumEven, sumOdd);  }  // Function to implement the recursive sum of even and odd numbers calculation  void q4(int start, int end) {  // Ensure that start is less than or equal to end  if (start > end) {  std::cout << "Invalid range." << std::endl;  return;  }  int sumEven = 0;  int sumOdd = 0;  // Call the sumEvenOddRecursive function  sumEvenOddRecursive(start, end, sumEven, sumOdd);  // Display the results  std::cout << "Sum of even numbers: " << sumEven << std::endl;  std::cout << "Sum of odd numbers: " << sumOdd << std::endl;  }  int main() {  // Example usage  int start, end |