**List of function programming exercises**

**1.** **Write a Pseudo code to find cube of any number using function.**

**Function Cube(Number):**

**// Calculate the cube of the given number**

**Result = Number \* Number \* Number**

**Return Result**

**// Main Program**

**Start**

**// Input a number from the user**

**Print "Enter a number:"**

**Input Number**

**// Call the Cube function and store the result**

**CubeResult = Cube(Number)**

**// Output the result**

**Print "The cube of", Number, "is", CubeResult**

**End**

**2.** **Write a Pseudo code to find diameter, circumference and area of circle using functions.**

**Function CalculateDiameter(Radius):**

**// Calculate the diameter of the circle**

**Diameter = 2 \* Radius**

**Return Diameter**

**Function CalculateCircumference(Radius):**

**// Calculate the circumference of the circle**

**Circumference = 2 \* PI \* Radius**

**Return Circumference**

**Function CalculateArea(Radius):**

**// Calculate the area of the circle**

**Area = PI \* Radius \* Radius**

**Return Area**

**// Main Program**

**Start**

**// Define the value of PI**

**PI = 3.14159**

**// Input the radius from the user**

**Print "Enter the radius of the circle:"**

**Input Radius**

**// Call the CalculateDiameter function and store the result**

**Diameter = CalculateDiameter(Radius)**

**// Call the CalculateCircumference function and store the result**

**Circumference = CalculateCircumference(Radius)**

**// Call the CalculateArea function and store the result**

**Area = CalculateArea(Radius)**

**// Output the results**

**Print "Diameter of the circle is", Diameter**

**Print "Circumference of the circle is", Circumference**

**Print "Area of the circle is", Area**

**End**

**3.** **Write a Pseudo code to find maximum and minimum between two numbers using functions.**

**Function FindMaximum(Number1, Number2):**

**// Determine the maximum of the two numbers**

**If Number1 > Number2 Then**

**Maximum = Number1**

**Else**

**Maximum = Number2**

**EndIf**

**Return Maximum**

**Function FindMinimum(Number1, Number2):**

**// Determine the minimum of the two numbers**

**If Number1 < Number2 Then**

**Minimum = Number1**

**Else**

**Minimum = Number2**

**EndIf**

**Return Minimum**

**// Main Program**

**Start**

**// Input two numbers from the user**

**Print "Enter the first number:"**

**Input Number1**

**Print "Enter the second number:"**

**Input Number2**

**// Call the FindMaximum function and store the result**

**Max = FindMaximum(Number1, Number2)**

**// Call the FindMinimum function and store the result**

**Min = FindMinimum(Number1, Number2)**

**// Output the results**

**Print "The maximum of", Number1, "and", Number2, "is", Max**

**Print "The minimum of", Number1, "and", Number2, "is", Min**

**End**

**4.** **Write a Pseudo code to check whether a number is even or odd using functions.**

**Function IsEven(Number):**

**// Check if the number is even**

**If Number MOD 2 = 0 Then**

**Return True**

**Else**

**Return False**

**EndIf**

**Function IsOdd(Number):**

**// Check if the number is odd**

**If Number MOD 2 ≠ 0 Then**

**Return True**

**Else**

**Return False**

**EndIf**

**// Main Program**

**Start**

**// Input a number from the user**

**Print "Enter a number:"**

**Input Number**

**// Call the IsEven function and check the result**

**If IsEven(Number) Then**

**Print Number, "is even"**

**Else**

**Print Number, "is odd"**

**EndIf**

**End**

**5.** **Write a Pseudo code to check whether a number is prime, Armstrong or perfect number using functions.**

**Function IsPrime(Number):**

**// Check if the number is prime**

**If Number <= 1 Then**

**Return False**

**EndIf**

**For i = 2 To sqrt(Number) Do**

**If Number MOD i = 0 Then**

**Return False**

**EndIf**

**EndFor**

**Return True**

**Function IsArmstrong(Number):**

**// Check if the number is an Armstrong number**

**Digits = CountDigits(Number)**

**Sum = 0**

**Temp = Number**

**While Temp > 0 Do**

**Digit = Temp MOD 10**

**Sum = Sum + Digit^Digits**

**Temp = Temp DIV 10**

**EndWhile**

**If Sum = Number Then**

**Return True**

**Else**

**Return False**

**EndIf**

**Function IsPerfect(Number):**

**// Check if the number is a perfect number**

**Sum = 0**

**For i = 1 To Number/2 Do**

**If Number MOD i = 0 Then**

**Sum = Sum + i**

**EndIf**

**EndFor**

**If Sum = Number Then**

**Return True**

**Else**

**Return False**

**EndIf**

**Function CountDigits(Number):**

**// Count the number of digits in the number**

**Digits = 0**

**Temp = Number**

**While Temp > 0 Do**

**Digits = Digits + 1**

**Temp = Temp DIV 10**

**EndWhile**

**Return Digits**

**// Main Program**

**Start**

**// Input a number from the user**

**Print "Enter a number:"**

**Input Number**

**// Call the IsPrime function and check the result**

**If IsPrime(Number) Then**

**Print Number, "is a prime number"**

**Else**

**Print Number, "is not a prime number"**

**EndIf**

**// Call the IsArmstrong function and check the result**

**If IsArmstrong(Number) Then**

**Print Number, "is an Armstrong number"**

**Else**

**Print Number, "is not an Armstrong number"**

**EndIf**

**// Call the IsPerfect function and check the result**

**If IsPerfect(Number) Then**

**Print Number, "is a perfect number"**

**Else**

**Print Number, "is not a perfect number"**

**EndIf**

**End**

**6.** **Write a Pseudo code to find all prime numbers between given interval using functions.**

**Function IsPrime(Number):**

**// Check if the number is prime**

**If Number <= 1 Then**

**Return False**

**EndIf**

**For i = 2 To sqrt(Number) Do**

**If Number MOD i = 0 Then**

**Return False**

**EndIf**

**EndFor**

**Return True**

**Function FindPrimesInRange(Start, End):**

**// Find all prime numbers in the interval [Start, End]**

**For Number = Start To End Do**

**If IsPrime(Number) Then**

**Print Number**

**EndIf**

**EndFor**

**// Main Program**

**Start**

**// Input the interval range from the user**

**Print "Enter the start of the interval:"**

**Input Start**

**Print "Enter the end of the interval:"**

**Input End**

**// Validate the interval range**

**If Start > End Then**

**Print "Invalid interval. The start should be less than or equal to the end."**

**Else**

**// Call the FindPrimesInRange function to print all primes in the given range**

**Print "Prime numbers between", Start, "and", End, "are:"**

**FindPrimesInRange(Start, End)**

**EndIf**

**End**

**7.** **Write a Pseudo code to print all strong numbers between given interval using functions.**

**Here's a pseudo code to find and print all strong numbers (also known as factorial sum numbers) within a given interval using functions:**

**Function Factorial(Number):**

**// Calculate the factorial of a given number**

**If Number = 0 Or Number = 1 Then**

**Return 1**

**EndIf**

**FactorialResult = 1**

**For i = 2 To Number Do**

**FactorialResult = FactorialResult \* i**

**EndFor**

**Return FactorialResult**

**Function IsStrongNumber(Number):**

**// Check if the number is a strong number**

**SumOfFactorials = 0**

**Temp = Number**

**While Temp > 0 Do**

**Digit = Temp MOD 10**

**Temp = Temp DIV 10**

**SumOfFactorials = SumOfFactorials + Factorial(Digit)**

**EndWhile**

**If SumOfFactorials = Number Then**

**Return True**

**Else**

**Return False**

**EndIf**

**Function FindStrongNumbersInRange(Start, End):**

**// Find and print all strong numbers in the interval [Start, End]**

**For Number = Start To End Do**

**If IsStrongNumber(Number) Then**

**Print Number**

**EndIf**

**EndFor**

**// Main Program**

**Start**

**// Input the interval range from the user**

**Print "Enter the start of the interval:"**

**Input Start**

**Print "Enter the end of the interval:"**

**Input End**

**// Validate the interval range**

**If Start > End Then**

**Print "Invalid interval. The start should be less than or equal to the end."**

**Else**

**// Call the FindStrongNumbersInRange function to print all strong numbers in the given range**

**Print "Strong numbers between", Start, "and", End, "are:"**

**FindStrongNumbersInRange(Start, End)**

**EndIf**

**End**

**8.** **Write a Pseudo code to print all Armstrong numbers between given interval using functions.**

**Function CountDigits(Number):**

**// Count the number of digits in the number**

**Digits = 0**

**Temp = Number**

**While Temp > 0 Do**

**Digits = Digits + 1**

**Temp = Temp DIV 10**

**EndWhile**

**Return Digits**

**Function IsArmstrong(Number):**

**// Check if the number is an Armstrong number**

**Digits = CountDigits(Number)**

**Sum = 0**

**Temp = Number**

**While Temp > 0 Do**

**Digit = Temp MOD 10**

**Sum = Sum + Digit^Digits**

**Temp = Temp DIV 10**

**EndWhile**

**If Sum = Number Then**

**Return True**

**Else**

**Return False**

**EndIf**

**Function FindArmstrongNumbersInRange(Start, End):**

**// Find and print all Armstrong numbers in the interval [Start, End]**

**For Number = Start To End Do**

**If IsArmstrong(Number) Then**

**Print Number**

**EndIf**

**EndFor**

**// Main Program**

**Start**

**// Input the interval range from the user**

**Print "Enter the start of the interval:"**

**Input Start**

**Print "Enter the end of the interval:"**

**Input End**

**// Validate the interval range**

**If Start > End Then**

**Print "Invalid interval. The start should be less than or equal to the end."**

**Else**

**// Call the FindArmstrongNumbersInRange function to print all Armstrong numbers in the given range**

**Print "Armstrong numbers between", Start, "and", End, "are:"**

**FindArmstrongNumbersInRange(Start, End)**

**EndIf**

**End**

**9.** **Write a Pseudo code to print all perfect numbers between given interval using functions.**

**Function SumOfDivisors(Number):**

**// Calculate the sum of all positive divisors of the number (excluding the number itself)**

**Sum = 0**

**For i = 1 To Number/2 Do**

**If Number MOD i = 0 Then**

**Sum = Sum + i**

**EndIf**

**EndFor**

**Return Sum**

**Function IsPerfectNumber(Number):**

**// Check if the number is a perfect number**

**If Number <= 1 Then**

**Return False**

**EndIf**

**If SumOfDivisors(Number) = Number Then**

**Return True**

**Else**

**Return False**

**EndIf**

**Function FindPerfectNumbersInRange(Start, End):**

**// Find and print all perfect numbers in the interval [Start, End]**

**For Number = Start To End Do**

**If IsPerfectNumber(Number) Then**

**Print Number**

**EndIf**

**EndFor**

**// Main Program**

**Start**

**// Input the interval range from the user**

**Print "Enter the start of the interval:"**

**Input Start**

**Print "Enter the end of the interval:"**

**Input End**

**// Validate the interval range**

**If Start > End Then**

**Print "Invalid interval. The start should be less than or equal to the end."**

**Else**

**// Call the FindPerfectNumbersInRange function to print all perfect numbers in the given range**

**Print "Perfect numbers between", Start, "and", End, "are:"**

**FindPerfectNumbersInRange(Start, End)**

**EndIf**

**End**

**10.** **Write a Pseudo code to find power of any number using recursion.**

**Function Power(Base, Exponent):**

**// Calculate Base^Exponent using recursion**

**If Exponent = 0 Then**

**Return 1**

**ElseIf Exponent = 1 Then**

**Return Base**

**Else**

**// Recursive case**

**Return Base \* Power(Base, Exponent - 1)**

**EndIf**

**// Main Program**

**Start**

**// Input the base and exponent from the user**

**Print "Enter the base:"**

**Input Base**

**Print "Enter the exponent:"**

**Input Exponent**

**// Call the Power function to compute the result**

**Result = Power(Base, Exponent)**

**// Output the result**

**Print Base, "raised to the power of", Exponent, "is", Result**

**End**

**11.** **Write a Pseudo code to print all natural numbers between 1 to n using recursion.**

**Function PrintNaturalNumbers(Current, n):**

**// Base case: If Current is greater than n, stop the recursion**

**If Current > n Then**

**Return**

**// Print the current number**

**Print Current**

**// Recursive case: Call the function with the next number**

**PrintNaturalNumbers(Current + 1, n)**

**// Main Program**

**Start**

**// Input the value of n from the user**

**Print "Enter the value of n:"**

**Input n**

**// Call the PrintNaturalNumbers function to print numbers from 1 to n**

**Print "Natural numbers from 1 to", n, "are:"**

**PrintNaturalNumbers(1, n)**

**End**

**12.** **Write a Pseudo code to print all even or odd numbers in given range using recursion.**

**Function to Print Even Numbers**

**Function PrintEvenNumbers(Current, End):**

**// Base case: If Current is greater than End, stop the recursion**

**If Current > End Then**

**Return**

**// Print the current even number**

**Print Current**

**// Recursive case: Call the function with the next even number**

**PrintEvenNumbers(Current + 2, End)**

**// Main Program for Even Numbers**

**Start**

**// Input the range from the user**

**Print "Enter the start of the range:"**

**Input Start**

**Print "Enter the end of the range:"**

**Input End**

**// Validate the range**

**If Start > End Then**

**Print "Invalid range. The start should be less than or equal to the end."**

**Else**

**// Adjust the start to be the next even number if it's odd**

**If Start MOD 2 ≠ 0 Then**

**Start = Start + 1**

**EndIf**

**// Call the PrintEvenNumbers function to print even numbers in the given range**

**Print "Even numbers between", Start, "and", End, "are:"**

**PrintEvenNumbers(Start, End)**

**EndIf**

**End**

**Function to Print Odd Numbers**

**Function PrintOddNumbers(Current, End):**

**// Base case: If Current is greater than End, stop the recursion**

**If Current > End Then**

**Return**

**// Print the current odd number**

**Print Current**

**// Recursive case: Call the function with the next odd number**

**PrintOddNumbers(Current + 2, End)**

**// Main Program for Odd Numbers**

**Start**

**// Input the range from the user**

**Print "Enter the start of the range:"**

**Input Start**

**Print "Enter the end of the range:"**

**Input End**

**// Validate the range**

**If Start > End Then**

**Print "Invalid range. The start should be less than or equal to the end."**

**Else**

**// Adjust the start to be the next odd number if it's even**

**If Start MOD 2 = 0 Then**

**Start = Start + 1**

**EndIf**

**// Call the PrintOddNumbers function to print odd numbers in the given range**

**Print "Odd numbers between", Start, "and", End, "are:"**

**PrintOddNumbers(Start, End)**

**EndIf**

**End**

**13.** **Write a Pseudo code to find sum of all natural numbers between 1 to n using recursion.**

**Function SumNaturalNumbers(n):**

**// Base case: If n is less than 1, the sum is 0**

**If n <= 0 Then**

**Return 0**

**// Recursive case: Sum of natural numbers up to n**

**Return n + SumNaturalNumbers(n - 1)**

**// Main Program**

**Start**

**// Input the value of n from the user**

**Print "Enter the value of n:"**

**Input n**

**// Call the SumNaturalNumbers function to compute the result**

**Result = SumNaturalNumbers(n)**

**// Output the result**

**Print "The sum of all natural numbers from 1 to", n, "is", Result**

**End**

**14.** **Write a Pseudo code to find sum of all even or odd numbers in given range.**

**Function to Sum Even Numbers**

**Function SumEvenNumbers(Current, End):**

**// Base case: If Current is greater than End, stop the recursion**

**If Current > End Then**

**Return 0**

**// If Current is even, include it in the sum**

**If Current MOD 2 = 0 Then**

**Return Current + SumEvenNumbers(Current + 2, End)**

**Else**

**// If Current is odd, skip it and continue with the next even number**

**Return SumEvenNumbers(Current + 1, End)**

**// Main Program for Even Numbers**

**Start**

**// Input the range from the user**

**Print "Enter the start of the range:"**

**Input Start**

**Print "Enter the end of the range:"**

**Input End**

**// Validate the range**

**If Start > End Then**

**Print "Invalid range. The start should be less than or equal to the end."**

**Else**

**// Adjust the start to be the next even number if it's odd**

**If Start MOD 2 ≠ 0 Then**

**Start = Start + 1**

**EndIf**

**// Call the SumEvenNumbers function to compute the sum of even numbers in the given range**

**Sum = SumEvenNumbers(Start, End)**

**Print "The sum of all even numbers between", Start, "and", End, "is", Sum**

**EndIf**

**End**

**Function to Sum Odd Numbers**

**Function SumOddNumbers(Current, End):**

**// Base case: If Current is greater than End, stop the recursion**

**If Current > End Then**

**Return 0**

**// If Current is odd, include it in the sum**

**If Current MOD 2 ≠ 0 Then**

**Return Current + SumOddNumbers(Current + 2, End)**

**Else**

**// If Current is even, skip it and continue with the next odd number**

**Return SumOddNumbers(Current + 1, End)**

**// Main Program for Odd Numbers**

**Start**

**// Input the range from the user**

**Print "Enter the start of the range:"**

**Input Start**

**Print "Enter the end of the range:"**

**Input End**

**// Validate the range**

**If Start > End Then**

**Print "Invalid range. The start should be less than or equal to the end."**

**Else**

**// Adjust the start to be the next odd number if it's even**

**If Start MOD 2 = 0 Then**

**Start = Start + 1**

**EndIf**

**// Call the SumOddNumbers function to compute the sum of odd numbers in the given range**

**Sum = SumOddNumbers(Start, End)**

**Print "The sum of all odd numbers between", Start, "and", End, "is", Sum**

**EndIf**

**End**

**15.** **Write a Pseudo code to find reverse of any number.**

**Function ReverseNumber(Number, Result = 0):**

**// Base case: If the number is 0, return the result**

**If Number = 0 Then**

**Return Result**

**// Extract the last digit of the number**

**LastDigit = Number MOD 10**

**// Update the result by appending the last digit**

**Result = Result \* 10 + LastDigit**

**// Remove the last digit from the number and call the function recursively**

**Return ReverseNumber(Number DIV 10, Result)**

**// Main Program**

**Start**

**// Input the number from the user**

**Print "Enter the number:"**

**Input Number**

**// Handle negative numbers by taking the absolute value**

**If Number < 0 Then**

**Number = -Number**

**// Call the ReverseNumber function to compute the reverse**

**ReversedNumber = ReverseNumber(Number)**

**// Output the reversed number**

**Print "The reverse of", Number, "is", ReversedNumber**

**End**

**16.** **Write a Pseudo code to check whether a number is palindrome or not.**

**Function ReverseNumber(Number, Result = 0):**

**// Base case: If Number is 0, return Result**

**If Number = 0 Then**

**Return Result**

**// Extract the last digit of Number**

**LastDigit = Number MOD 10**

**// Update Result by appending the last digit**

**Result = Result \* 10 + LastDigit**

**// Remove the last digit from Number and call the function recursively**

**Return ReverseNumber(Number DIV 10, Result)**

**// Function to check if a number is a palindrome**

**Function IsPalindrome(Number):**

**// Handle negative numbers as non-palindromes**

**If Number < 0 Then**

**Return False**

**// Call ReverseNumber to get the reversed number**

**ReversedNumber = ReverseNumber(Number)**

**// Check if the original number is equal to the reversed number**

**If Number = ReversedNumber Then**

**Return True**

**Else**

**Return False**

**// Main Program**

**Start**

**// Input the number from the user**

**Print "Enter the number:"**

**Input Number**

**// Call the IsPalindrome function to check if the number is a palindrome**

**If IsPalindrome(Number) Then**

**Print Number, "is a palindrome."**

**Else**

**Print Number, "is not a palindrome."**

**EndIf**

**End**

**17.** **Write a Pseudo code to find sum of digits of a given number.**

**Function SumOfDigits(Number):**

**// Base case: If Number is 0, the sum of digits is 0**

**If Number = 0 Then**

**Return 0**

**// Extract the last digit of the number**

**LastDigit = Number MOD 10**

**// Call SumOfDigits recursively with the remaining digits**

**Return LastDigit + SumOfDigits(Number DIV 10)**

**// Main Program**

**Start**

**// Input the number from the user**

**Print "Enter the number:"**

**Input Number**

**// Handle negative numbers by taking the absolute value**

**If Number < 0 Then**

**Number = -Number**

**// Call the SumOfDigits function to compute the sum of digits**

**Sum = SumOfDigits(Number)**

**// Output the result**

**Print "The sum of the digits of", Number, "is", Sum**

**End**

**18.** **Write a Pseudo code to find factorial of any number.**

**Function Factorial(Number):**

**// Base case: Factorial of 0 or 1 is 1**

**If Number = 0 OR Number = 1 Then**

**Return 1**

**// Recursive case: Factorial of Number is Number \* Factorial(Number - 1)**

**Return Number \* Factorial(Number - 1)**

**// Main Program**

**Start**

**// Input the number from the user**

**Print "Enter the number:"**

**Input Number**

**// Handle invalid inputs (negative numbers) for factorial**

**If Number < 0 Then**

**Print "Factorial is not defined for negative numbers."**

**Else**

**// Call the Factorial function to compute the result**

**Result = Factorial(Number)**

**// Output the result**

**Print "The factorial of", Number, "is", Result**

**EndIf**

**End**

**19.** **Write a Pseudo code to generate nth Fibonacci term.**

**Function Fibonacci(n):**

**// Base case: If n is 0, return 0**

**If n = 0 Then**

**Return 0**

**// Base case: If n is 1, return 1**

**If n = 1 Then**

**Return 1**

**// Recursive case: Return the sum of Fibonacci(n-1) and Fibonacci(n-2)**

**Return Fibonacci(n - 1) + Fibonacci(n - 2)**

**// Main Program**

**Start**

**// Input the value of n from the user**

**Print "Enter the value of n:"**

**Input n**

**// Validate the input to ensure it is non-negative**

**If n < 0 Then**

**Print "Fibonacci number is not defined for negative indices."**

**Else**

**// Call the Fibonacci function to compute the nth Fibonacci number**

**Result = Fibonacci(n)**

**// Output the result**

**Print "The", n, "th Fibonacci number is", Result**

**EndIf**

**End**

**20.** **Write a Pseudo code to display all array elements using recursion.**

**Function PrintArrayElements(Array, Index):**

**// Base case: If Index is equal to the length of the array, stop the recursion**

**If Index >= Length(Array) Then**

**Return**

**// Print the current element**

**Print Array[Index]**

**// Recursive case: Call the function with the next index**

**PrintArrayElements(Array, Index + 1)**

**// Main Program**

**Start**

**// Define the array and its length**

**Array = [/\* initialize with elements \*/]**

**LengthOfArray = Length(Array)**

**// Start printing array elements from index 0**

**Print "Array elements are:"**

**PrintArrayElements(Array, 0)**

**End**

**21.** **Write a Pseudo code to find sum of elements of array.**

**Function SumArrayElements(Array, Index):**

**// Base case: If Index is equal to the length of the array, return 0**

**If Index >= Length(Array) Then**

**Return 0**

**// Recursive case: Return the sum of the current element and the sum of the rest of the array**

**Return Array[Index] + SumArrayElements(Array, Index + 1)**

**// Main Program**

**Start**

**// Define the array and its length**

**Array = [/\* initialize with elements \*/]**

**LengthOfArray = Length(Array)**

**// Call the SumArrayElements function to compute the sum of array elements**

**TotalSum = SumArrayElements(Array, 0)**

**// Output the result**

**Print "The sum of array elements is", TotalSum**

**End**

**22.** **Write a Pseudo code to find maximum and minimum elements in array.**

**Function FindMax(Array, Index, CurrentMax):**

**// Base case: If Index is equal to the length of the array, return CurrentMax**

**If Index >= Length(Array) Then**

**Return CurrentMax**

**// Update CurrentMax if the current element is greater than CurrentMax**

**If Array[Index] > CurrentMax Then**

**CurrentMax = Array[Index]**

**// Recursive case: Call the function with the next index and updated CurrentMax**

**Return FindMax(Array, Index + 1, CurrentMax)**

**// Function to Find Minimum Element**

**Function FindMin(Array, Index, CurrentMin):**

**// Base case: If Index is equal to the length of the array, return CurrentMin**

**If Index >= Length(Array) Then**

**Return CurrentMin**

**// Update CurrentMin if the current element is less than CurrentMin**

**If Array[Index] < CurrentMin Then**

**CurrentMin = Array[Index]**

**// Recursive case: Call the function with the next index and updated CurrentMin**

**Return FindMin(Array, Index + 1, CurrentMin)**

**// Main Program**

**Start**

**// Define the array and its length**

**Array = [/\* initialize with elements \*/]**

**LengthOfArray = Length(Array)**

**// Initialize CurrentMax and CurrentMin with the first element of the array**

**InitialMax = Array[0]**

**InitialMin = Array[0]**

**// Call the FindMax function to compute the maximum element**

**MaxElement = FindMax(Array, 1, InitialMax)**

**// Call the FindMin function to compute the minimum element**

**MinElement = FindMin(Array, 1, InitialMin)**

**// Output the results**

**Print "The maximum element in the array is", MaxElement**

**Print "The minimum element in the array is", MinElement**

**End**