Easy Tasks (60-74 points)

**Task 1: Parity Checker**

**Objective:** Write a program to determine the parity of a given number.

* **Input:**
  + Integer: **25**
* **Output:**
  + Parity: **Odd**

**Instructions:** Implement a function that accepts an integer and returns its parity as "Even" or "Odd". Explain the logic and algorithm used in your implementation.

**Task 2: Prime Number Checker**

**Objective:** Implement a program that checks if a given number is prime.

* **Input:**
  + Integer: **17**
* **Output:**
  + Prime Status: **Prime**

**Instructions:** Develop a function that takes an integer input and returns whether it is "Prime" or "Not Prime". Provide an explanation of the method used to determine the primality.

**Task 3: GCD Calculator**

**Objective:** Create a program to calculate the Greatest Common Divisor (GCD) of two numbers.

* **Input:**
  + Two integers: **48, 18**
* **Output:**
  + GCD: **6**

**Instructions:** Implement a function that calculates the GCD of two integers. Explain the algorithm and steps involved in the calculation.

Medium Tasks (75-89 points)

**Task 4: Prime Factorization**

**Objective:** Write a program to find the prime factorization of a given number.

* **Input:**
  + Integer: **56**
* **Output:**
  + Prime Factors: **2^3 \* 7**

**Instructions:** Implement a function to find the prime factors and their powers. The program should provide the result in the format "p1^e1 \* p2^e2 \* ...".

**Task 5: LCM Calculator**

**Objective:** Create a program to compute the Least Common Multiple (LCM) of two integers.

* **Input:**
  + Two integers: **15, 20**
* **Output:**
  + LCM: **60**

**Instructions:** Develop a function to calculate the LCM, and explain the method used for the calculation and how it relates to the GCD.

**Task 6: Direct Proof Implementation**

**Objective:** Write a program that demonstrates a direct proof method for a given mathematical statement.

* **Input:**
  + Statement: “If a number is even, then it is divisible by 2.”
* **Output:**
  + Proof steps and validation

**Instructions:** Implement an algorithm that validates the statement through direct proof. Explain each step and the logical reasoning behind it.

Hard Tasks (90-100 points)

**Task 7: Advanced Number Theory Function**

**Objective:** Implement a program that performs a complex operation involving number theory concepts, like Euler’s Totient Function.

* **Input:**
  + Integer: **12**
* **Output:**
  + Euler’s Totient Value: **4**

**Instructions:** Implement Euler’s Totient Function and provide an explanation of the algorithm and steps in the computation.

**Task 8: Comprehensive Proof by Induction**

**Objective:** Develop a program that can prove mathematical statements using the principle of mathematical induction.

* **Input:**
  + Statement: “The sum of the first n odd numbers is n^2 for all positive integers n.”
* **Output:**
  + Proof steps and validation

**Instructions:** The program should facilitate proof by induction, demonstrating base case verification, induction hypothesis, and induction step.

**Task 9: Advanced Function Evaluation**

**Objective:** Implement a program to evaluate complex mathematical functions with considerations to their domains and ranges.

* **Input:**
  + Function: **f(x) = x^2 + 2x + 1**
  + Value: **x = 3**
* **Output:**
  + Evaluated Result: **16**

**Instructions:** The program should evaluate the function, consider its domain and range, and provide an explanation of the evaluation process and results.