Problem #1: Prime Number Function

- **Pseudo Code:**

```
FUNCTION isPrime(number)
   IF number < 2 THEN
      RETURN false
   END IF
   IF number == 2 THEN
      RETURN true
   END IF
   IF number MOD 2 == 0 THEN
      RETURN false
   END IF
   SET half TO number / 2
   FOR each divisor FROM 3 TO half IN STEPS OF 2
      IF number MOD divisor == 0 THEN
         RETURN false
      END IF
   END FOR
   RETURN true
END FUNCTION
```

- **Type Of Function:** Pure function. It returns a boolean indicating whether the input number is prime. A pure function was chosen because the output depends solely on the input, with no side effects or external state dependencies.
- **Parameters:** number (Int) - the natural number to check for primality.
- **Variables:** half (Int) - half of the number, used to reduce the range of numbers to check for divisors.
- **divisor (Int):** used in the loop to check if number has any divisors other than 1 and itself.
- **Constants:** No explicit constants are used in this function, but the parameter number acts like a constant within the function scope as it does not change.

Problem #2: Greatest Common Denominator

- **Pseudo Code:**

```
FUNCTION gcd(a, b)
   SET remainder TO a MOD b
   IF remainder != 0 THEN
      RETURN gcd(b, remainder)
   ELSE
```

```
    RETURN b
  END IF
END FUNCTION
```

- **Type of Function:** Recursive function. It calculates the greatest common divisor of two integers using recursion, following the Euclidean algorithm.
- **Parameters:**
    - (Int) - first integer.
    - (Int) - second integer.
- **Variables:** r (Int) - the remainder of a divided by b, used to recursively find the GCD.
- **Constants:** No explicit constants are defined, but a and b are treated as constants in each call as their values are passed without modification to subsequent recursive calls.

Problem #3: Verify Parenthesis
- **Pseudo Code:**

```
FUNCTION verifyParenthesis
  FOR each char IN expression
    IF char == "(" THEN
      PUSH char onto stack
    ELSE IF char == ")" THEN
      IF stack is empty THEN
        RETURN false
      END IF
      POP from stack
    END IF
  END FOR
  RETURN stack is empty
END FUNCTION
```

- **Type of Function:** Stack-based function. It checks if the parentheses in a string are correctly paired using a stack to track open parentheses.
- **Parameters:** expression, String - the string containing parentheses to check.
- **Variables:** stack, Character - a stack, implemented as an array, to hold opening parentheses found in the expression.
- **char (Character):** used in the loop to inspect each character in expression.
- **Constants:** No explicit constants are used. The parameter expression acts like a constant within the function as its value does not change.

Problem #4: Sum of Powers

- **Pseudo Code:**

```
FUNCTION sumOfPowers(n, m)
  SET sum TO 0
  FOR i FROM 1 TO m
    SET power TO 1
    FOR j FROM 1 TO n
      power = power * i
    END FOR
    sum = sum + power
  END FOR
  RETURN sum
END FUNCTION
```

- **Type of Function:** Iterative function. It computes the sum of powers from 1 to m, each raised to the power of n.
- **Parameters:**
  - n (Int) - the exponent used in the power calculation.
  - m (Int) - the upper limit of the base numbers in the summation.
- **Variables:**
  - sum (Int) - accumulates the sum of each base number raised to n.
  - power (Int) - stores the result of raising a base number to n, reset for each base number.
  - i (Int) - iterator used in the outer loop to go through each base number from 1 to m.
- **Constants:** No explicit constants are defined. The parameters n and m act as constants within the function scope, as their values do not change after being set.