All Matrix:

```
START PROGRAM
DECLARE 2D array matrix WITH VALUES {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}

OUTPUT "Is scalar matrix? " + CALL isScalarMatrix(matrix)
OUTPUT "Is triangular matrix? " + CALL isTriangularMatrix(matrix)
OUTPUT "Is diagonal matrix? " + CALL isDiagonalMatrix(matrix)
OUTPUT "Is symmetric matrix? " + CALL isSymmetricMatrix(matrix)
OUTPUT "Is asymmetric matrix? " + CALL isAsymmetricMatrix(matrix)
OUTPUT "Is idempotent matrix? " + CALL isIdempotentMatrix(matrix)
OUTPUT "Is square matrix? " + CALL isSquareMatrix(matrix)
OUTPUT "Is Hermitian matrix? " + CALL isHermitianMatrix(matrix)
OUTPUT "Is periodic matrix? " + CALL isPeriodicMatrix(matrix)
OUTPUT "Is nilpotent matrix? " + CALL isNilpotentMatrix(matrix)
OUTPUT "Is zero matrix? " + CALL isZeroMatrix(matrix)

FUNCTION isScalarMatrix(matrix)
   DECLARE scalar = matrix[0][0]
   FOR EACH row IN matrix
      FOR EACH element IN row
         IF element ≠ scalar THEN
            RETURN FALSE
         END IF
      END FOR
   END FOR
   RETURN TRUE
END FUNCTION

FUNCTION isTriangularMatrix(matrix)
   FOR i = 1 TO matrix.length - 1
```

```
        FOR j = 0 TO i - 1
            IF matrix[i][j] ≠ 0 THEN
                RETURN FALSE
            END IF
        END FOR
    END FOR
    RETURN TRUE
END FUNCTION

FUNCTION isDiagonalMatrix(matrix)
    FOR i = 0 TO matrix.length - 1
        FOR j = 0 TO matrix[0].length - 1
            IF i ≠ j AND matrix[i][j] ≠ 0 THEN
                RETURN FALSE
            END IF
        END FOR
    END FOR
    RETURN TRUE
END FUNCTION

FUNCTION isSymmetricMatrix(matrix)
    FOR i = 0 TO matrix.length - 1
        FOR j = 0 TO matrix[0].length - 1
            IF matrix[i][j] ≠ matrix[j][i] THEN
                RETURN FALSE
            END IF
        END FOR
    END FOR
    RETURN TRUE
END FUNCTION
```

```
FUNCTION isAsymmetricMatrix(matrix)
    RETURN NOT isSymmetricMatrix(matrix)
END FUNCTION

FUNCTION isIdempotentMatrix(matrix)
    DECLARE result = CALL multiplyMatrices(matrix, matrix)
    FOR i = 0 TO matrix.length - 1
        FOR j = 0 TO matrix[0].length - 1
            IF matrix[i][j] ≠ result[i][j] THEN
                RETURN FALSE
            END IF
        END FOR
    END FOR
    RETURN TRUE
END FUNCTION

FUNCTION isSquareMatrix(matrix)
    RETURN matrix.length = matrix[0].length
END FUNCTION

FUNCTION isHermitianMatrix(matrix)
    RETURN isSymmetricMatrix(matrix)
END FUNCTION

FUNCTION isPeriodicMatrix(matrix)
    DECLARE identity = CALL createIdentityMatrix(matrix.length)
    DECLARE power = CALL multiplyMatrices(matrix, matrix)
    DECLARE result = CALL multiplyMatrices(power, matrix)
    FOR i = 0 TO matrix.length - 1
        FOR j = 0 TO matrix[0].length - 1
            IF identity[i][j] ≠ result[i][j] THEN
```

```
                RETURN FALSE
            END IF
        END FOR
    END FOR
    RETURN TRUE
END FUNCTION

FUNCTION isNilpotentMatrix(matrix)
    DECLARE power = CALL multiplyMatrices(matrix, matrix)
    IF CALL isZeroMatrix(power) THEN
        RETURN TRUE
    END IF
    DECLARE result = CALL multiplyMatrices(power, matrix)
    RETURN CALL isZeroMatrix(result)
END FUNCTION

FUNCTION isZeroMatrix(matrix)
    FOR EACH row IN matrix
        FOR EACH element IN row
            IF element ≠ 0 THEN
                RETURN FALSE
            END IF
        END FOR
    END FOR
    RETURN TRUE
END FUNCTION

FUNCTION multiplyMatrices(matrix1, matrix2)
    DECLARE rows1 = matrix1.length
    DECLARE cols1 = matrix1[0].length
    DECLARE cols2 = matrix2[0].length
    DECLARE 2D array result WITH DIMENSIONS rows1 x cols2
    FOR i = 0 TO rows1 - 1
        FOR j = 0 TO cols2 - 1
            FOR k = 0 TO cols1 - 1
                result[i][j] += matrix1[i][k] * matrix2[k][j]
            END FOR
        END FOR
    END FOR
    RETURN result
END FUNCTION

FUNCTION createIdentityMatrix(size)
    DECLARE 2D array identity WITH DIMENSIONS size x size
    FOR i = 0 TO size - 1
        identity[i][i] = 1
    END FOR
    RETURN identity
END FUNCTION

END PROGRAM
```

Addition of Two Matrix:

```
START PROGRAM

DECLARE 2D array a WITH VALUES {{2, 1, 9}, {4, 2, 4}, {0, -6, 2}}
DECLARE 2D array b WITH VALUES {{9, 1, 5}, {2, 1, 8}, {11, 4, 3}}
DECLARE 2D array c WITH DIMENSIONS 3 x 3

FOR i = 0 TO 2
    FOR j = 0 TO 2
        c[i][j] = a[i][j] + b[i][j]
        OUTPUT c[i][j] + " "
    END FOR
    OUTPUT NEW LINE
END FOR

END PROGRAM
```

Subtraction of Two Matrix:

```
START PROGRAM

FUNCTION printMatrix(M, rowSize, colSize)
    FOR i = 0 TO rowSize - 1
        FOR j = 0 TO colSize - 1
            OUTPUT M[i][j] + " "
        END FOR
        OUTPUT NEW LINE
    END FOR
END FUNCTION

FUNCTION subtract(A, B, size)
    DECLARE 2D array C WITH DIMENSIONS size x size
    FOR i = 0 TO size - 1
        FOR j = 0 TO size - 1
            C[i][j] = A[i][j] - B[i][j]
        END FOR
    END FOR
    RETURN C
END FUNCTION
```

```
DECLARE size = 3
DECLARE 2D array A WITH VALUES {{2, 1, 9}, {4, 2, 4}, {0, -6, 2}}
DECLARE 2D array B WITH VALUES {{9, 1, 5}, {2, 1, 8}, {11, 4, 3}}

OUTPUT "\nMatrix A:"
CALL printMatrix(A, size, size)

OUTPUT "\nMatrix B:"
CALL printMatrix(B, size, size)

DECLARE 2D array C = CALL subtract(A, B, size)

OUTPUT "\nResultant Matrix:"
CALL printMatrix(C, size, size)

END PROGRAM
```

Multiply Two Matrix:

```
START PROGRAM

DECLARE 2D array a WITH VALUES {{2, 1, 9}, {4, 2, 4}, {0, -6, 2}}
DECLARE 2D array b WITH VALUES {{9, 1, 5}, {2, 1, 8}, {11, 4, 3}}
DECLARE 2D array c WITH DIMENSIONS 3 x 3

FOR i = 0 TO 2
   FOR j = 0 TO 2
      c[i][j] = 0
      FOR k = 0 TO 2
         c[i][j] += a[i][k] * b[k][j]
```

```
        END FOR
        OUTPUT c[i][j] + " "
    END FOR
    OUTPUT NEW LINE
END FOR

END PROGRAM
```

Binary Search Algorithms

```
FUNCTION binarySearch(arr, target)
    left ← 0
    right ← length(arr) - 1

    WHILE left <= right
        mid ← left + (right - left) / 2

        IF arr[mid] = target
            RETURN mid
        ELSE IF arr[mid] < target
            left ← mid + 1
        ELSE
            right ← mid - 1
        END IF
    END WHILE

    RETURN -1
END FUNCTION

PROGRAM Main
```

```
    arr ← [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]
    target ← 23
    index ← binarySearch(arr, target)

    IF index = -1
        OUTPUT "Target not found in the array."
    ELSE
        OUTPUT "Target found at index: ", index
    END IF
END PROGRAM
```

Linear Search Algorithms:

```
FUNCTION linearSearch(arr, target)
    FOR i FROM 0 TO length(arr) - 1
        IF arr[i] = target
            RETURN i
        END IF
    END FOR
    RETURN -1
END FUNCTION

PROGRAM Main
    arr ← [5, 2, 8, 12, 16, 23, 38, 56, 72, 91]
    target ← 23
    index ← linearSearch(arr, target)

    IF index = -1
        OUTPUT "Target not found in the array."
    ELSE
```

```
        OUTPUT "Target found at index: ", index
    END IF
END PROGRAM
```