**NAME = Suraj Vishwakarma,**
**QID - 24510031**
**MCA 2nd Year**

# Week 5 Practice Sheet

## SECTION A: Coding Problems

### Q1. Write a program to represent a matrix using 2D arrays and display it.

**Answer:**

```
public class MatrixDisplay {
    public static void main(String[]
args) {          int[][] matrix = {
{1, 2, 3},
             {4, 5, 6},
             {7, 8, 9}
         };

         System.out.println("Matrix:");          for
(int i = 0; i < matrix.length; i++) {
for (int j = 0; j < matrix[0].length; j++) {
                 System.out.print(matrix[i][j] + " ");
             }
             System.out.println();
         }
    }
}
```

### Q2. Write a program to take input of an n × m matrix and print it in matrix format.

**Answer:**

```
import java.util.Scanner;

public class InputMatrix {
    public static void main(String[]
args) {          Scanner sc = new
Scanner(System.in);
System.out.print("Enter rows: ");
int n = sc.nextInt();
         System.out.print("Enter columns:
");          int m = sc.nextInt();
          int[][] matrix = new int[n][m];
System.out.println("Enter elements:");
for (int i = 0; i < n; i++) {
for (int j = 0; j < m; j++) {
matrix[i][j] = sc.nextInt();
             }
         }
```

```
System.out.println("Matrix:");
for (int i = 0; i < n; i++) {
for (int j = 0; j < m; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

---

## Q3. Write a program to find the transpose of a matrix.

### Answer:

```
public class TransposeMatrix {
    public static void main(String[]
args) {          int[][] matrix = {
            {1, 2, 3},
            {4, 5, 6}
        };           int rows =
matrix.length;          int cols
= matrix[0].length;
        int[][] transpose = new int[cols][rows];
         for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
transpose[j][i] = matrix[i][j];
            }
        }

        System.out.println("Transpose:");
for (int i = 0; i < cols; i++) {
for (int j = 0; j < rows; j++) {
                System.out.print(transpose[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

---

## Q4. Write a program to rotate a square matrix by 90° clockwise.

### Answer:

```
public class RotateClockwise {
    public static void main(String[]
args) {          int[][] matrix = {
{1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };
        int n = matrix.length;

        // Transpose
```

```
        for (int i = 0; i < n; i++) {
for (int j = i; j < n; j++) {
int temp = matrix[i][j];
matrix[i][j] = matrix[j][i];
matrix[j][i] = temp;
            }
        }

        // Reverse rows
        for (int i = 0; i < n; i++) {
for (int j = 0; j < n / 2; j++) { int
temp = matrix[i][j]; matrix[i][j] =
matrix[i][n - 1 - j]; matrix[i][n - 1
- j] = temp;
            }
        }

        System.out.println("90° Clockwise Rotation:");
for (int[] row : matrix) {
            for (int val : row) System.out.print(val + " ");
            System.out.println();
        }
    }
}
```

---

## Q5. Write a program to rotate a square matrix by 90° anticlockwise.

**Answer:**

```
public class RotateAnticlockwise { public
static void main(String[] args) { int[][]
matrix = {                      {1, 2,
3},
            {4, 5, 6},
            {7, 8, 9}
        };
        int n = matrix.length;

        // Transpose
        for (int i = 0; i < n; i++) {
for (int j = i; j < n; j++) {
int temp = matrix[i][j];
matrix[i][j] = matrix[j][i];
matrix[j][i] = temp;
            }
        }

        // Reverse columns          for
(int j = 0; j < n; j++) {               for
(int i = 0; i < n / 2; i++) {
int temp = matrix[i][j];
matrix[i][j] = matrix[n - 1 - i][j];
matrix[n - 1 - i][j] = temp;
            }
        }
```

```
        System.out.println("90° Anticlockwise Rotation:");
for (int[] row : matrix) {
            for (int val : row) System.out.print(val + " ");
            System.out.println();
        }

} }
```

## Q6. Write a program to check whether a square matrix is symmetric ($A = A^T$).

**Answer:**

```
public class SymmetricMatrix {
    public static void main(String[]
args) {          int[][] matrix = {
{1, 2, 3},
             {2, 5, 6},
             {3, 6, 9}
         };
         int n = matrix.length;
boolean isSymmetric = true;
          for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
if (matrix[i][j] != matrix[j][i]) {
isSymmetric = false;                        break;
                 }
             }

}                                          if (isSymmetric)
                                               System.out.println("Matrix is

                                               Symmetric");
else
                                               System.out.println("Matrix is NOT
    }
}                                              Symmetric");
```

---

## Q7. Write a program to multiply two matrices (if dimensions allow).

**Answer:**

```
public class MatrixMultiplication {
public static void main(String[] args) {
int[][] A = {              {1, 2, 3},
             {4, 5, 6}
         };
int[][] B = {
{7, 8},
             {9, 10},
             {11, 12}
         };          int rows =
A.length;          int cols =
B[0].length;          int sumLength =
B.length;          int[][] C = new
int[rows][cols];
```

```
          for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
for (int k = 0; k < sumLength; k++) {
                    C[i][j] += A[i][k] * B[k][j];
                }
            }
        }

        System.out.println("Matrix Multiplication Result:");
for (int[] row : C) {
            for (int val : row) System.out.print(val + " ");
            System.out.println();
        }
    }
}
```

---

## Q8. Write a program to add two matrices.

**Answer:**

```
public class MatrixAddition {
    public static void main(String[]
args) {          int[][] A = {
{1, 2},
            {3,
4}          };
int[][] B = {
{5, 6},
            {7, 8}
        };           int rows =
A.length;          int cols =
A[0].length;          int[][] C = new
int[rows][cols];
          for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
C[i][j] = A[i][j] + B[i][j];
            }
        }

        System.out.println("Matrix Addition Result:");
for (int[] row : C) {
            for (int val : row) System.out.print(val + " ");
System.out.println();
        }
    }
}
```

---

## Q9. Write a program to convert a matrix into sparse matrix representation (row, col, value triplets).

**Answer:**

```
public class SparseMatrix {
```

```
    public static void main(String[]
args) {          int[][] matrix = {
{0, 0, 3},
             {4, 0, 0},
             {0, 5, 0}
         };

         System.out.println("Sparse Matrix Representation (row, col,
value):");
         for (int i = 0; i < matrix.length; i++) {
for (int j = 0; j < matrix[0].length; j++) {
                 if (matrix[i][j] != 0) {
                     System.out.println(i + " " + j + " " + matrix[i][j]);
                 }
             }
         }
    }
}
```

---

## Q10. Write a program to perform addition of two sparse matrices.

**Answer:**

```
{4, 0, 0},
{0, 5, 0}

{

{0, 0, 6},
{7, 0, 0}

public class SparseMatrixAddition {
public static void main(String[] args) {
int[][] A = {              {0, 0, 3},


         };
int[][] B =
{0, 2, 0},


         };          int rows = A.length,
cols = A[0].length;          int[][] result =
new int[rows][cols];

         for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
result[i][j] = A[i][j] + B[i][j];
             }
         }

         System.out.println("Sparse Matrix Addition Result (row, col,
value):");
         for (int i = 0; i < rows; i++) {
for (int j = 0; j < cols; j++) {
if (result[i][j] != 0) {
                     System.out.println(i + " " + j + " " + result[i][j]);
                 }
             }
         }
    }
```

```
}
```

## Q11. Write a program to find the sum of diagonal elements of a square matrix.

**Answer:**

```
public class DiagonalSum {
    public static void main(String[]
args) {          int[][] matrix = {
{1, 2, 3},
             {4, 5, 6},
             {7, 8, 9}

};
         int sum = 0;
         for (int i = 0; i < matrix.length; i++) {
sum += matrix[i][i];
         }

         System.out.println("Sum of Diagonal Elements: " + sum);
    }
}
```

---

## Q12. Write a program to check if a given matrix is an identity matrix.

**Answer:**

```
public class IdentityMatrix {
    public static void main(String[]
args) {          int[][] matrix = {
{1, 0, 0},
             {0, 1, 0},
             {0, 0, 1}
         };           boolean
isIdentity = true;          int
n = matrix.length;
         for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
                 if ((i == j && matrix[i][j] != 1) || (i != j &&
matrix[i][j]
!= 0)) {                                        isIdentity = false;

break;                                      }

             }

}                                           if (isIdentity)
                                                System.out.println("Matrix is an

else                                                Identity Matrix");

                                                System.out.println("Matrix is NOT
    }
}                                               an Identity Matrix");
```

---

## Q13. Write a program to find the maximum element in a matrix.

**Answer:**

```
public class MaxElement {
    public static void main(String[]
args) {          int[][] matrix = {
{1, 20, 3},
            {14, 5, 6},
            {7, 8, 9}
        };           int max =
matrix[0][0];          for (int[] row
: matrix) {              for (int val
: row) {                  if (val >
max) max = val;
            }
        }

        System.out.println("Maximum Element: " + max);
    }
}
```

---

## Q14. Write a program to count total number of zeros in a matrix.

**Answer:**

```
public class CountZeros {
    public static void main(String[]
args) {          int[][] matrix = {
{0, 2, 0},
            {4, 0, 6},
            {0, 0, 9}
        };
        int count = 0;          for
(int[] row : matrix) {
for (int val : row) {
if (val == 0) count++;
            }
        }

        System.out.println("Total number of zeros: " + count);
    }
}
```

---

## Q15. Write a program to search an element in a row-wise & column-wise sorted matrix.

**Answer:**

```
public class SearchSortedMatrix {
public static void main(String[] args) {
int[][] matrix = {              {1, 4, 7,
11},
            {2, 5, 8, 12},
            {3, 6, 9, 16},
            {10, 13, 14, 17}
        };           int
target = 14;          int n =
```

```
matrix.length;          int m
= matrix[0].length;
         int i = 0, j = m
- 1;          boolean found
= false;

        while (i < n && j >= 0) {
if (matrix[i][j] == target) {
found = true;                   break;
            } else if (matrix[i][j] >
target) {                    j--;
} else {                    i++;
            }

}                                                   if (found)
                                                        System.out.println("Element " +
                                                        target + " found.");

else
                                                        System.out.println("Element " +
                                                        target + " not

found.");      }
}
```

# SECTION B:

**Q1. Which of the following is the correct way to represent a matrix in memory?** a) Row-major order
b) Column-major order
c) Both (depends on language)
d) None
**Answer: (c) Both (depends on language)**

---

**Q2. In C/C++, matrices are stored in:** a) Column-major order
b) Row-major order
c) Linked list format
d) Hash table
**Answer: (b) Row-major order**

---

**Q3. What is the space complexity of storing an n × m matrix in a 2D array?** a) O(1)
b) O(n)
c) O(m)
d) O(n × m)
**Answer: (d) O(n × m)**

---

**Q4. A square matrix means:** a) Rows ≠ Columns
b) Rows = Columns
c) Only diagonal elements exist
d) Matrix is symmetric
**Answer: (b) Rows = Columns**

---

**Q5. Which of the following is a diagonal matrix?** a) All elements are 0
b) All non-diagonal elements are 0
c) All diagonal elements are 0
d) All elements are 1
**Answer: (b) All non-diagonal elements are 0**

---

**Q6. An identity matrix is:** a) All 0s
b) All 1s
c) Diagonal elements = 1, others = 0
d) Diagonal elements = 0, others = 1
**Answer: (c) Diagonal elements = 1, others = 0**

---

**Q7. Which of the following cannot be performed on matrices directly?** a) Addition
b) Subtraction
c) Division
d) Multiplication **Answer: (c) Division**

---

**Q8. Matrix addition is possible if:** a) Rows and columns are equal
b) Only rows equal
c) Only columns equal
d) None
**Answer: (a) Rows and columns are equal**

---

**Q9. Matrix multiplication is possible if:** a) Columns of first = Rows of second
b) Rows of first = Columns of second
c) Both dimensions equal
d) Only square matrices allowed
**Answer: (a) Columns of first = Rows of second**

---

**Q10. What is the order of the product matrix C = A × B if A is m × n and B is n × p?** a) m × n
b) n × p
c) m × p
d) p × m
**Answer: (c) m × p**

---

## Matrix Operations (Transpose, Rotation, Multiplication)

**Q11. Transpose of a matrix means:** a) Interchanging rows and columns
b) Doubling each element
c) Squaring elements
d) Multiplying diagonals
**Answer: (a) Interchanging rows and columns**

---

**Q12. The transpose of transpose of a matrix A is:** a) -A
b) A
c) $A^2$
d) None **Answer: (b) A**

---

**Q13. If A is a symmetric matrix, then:** a) $A = -A^T$
b) $A = A^T$
c) $A^2 = I$
d) $A = A^{-1}$ **Answer: (b) $A = A^T$**

---

**Q14. A skew-symmetric matrix satisfies:** a) $A = A^T$

b) $A = -A^T$
c) $A = I$
d) $A = A^2$
**Answer: (b) $A = -A^T$**

---

**Q15. Time complexity of matrix multiplication (naive method):** a) $O(n^2)$
b) $O(n^3)$
c) $O(n \log n)$
d) $O(n^2 \log n)$
**Answer: (b) $O(n^3)$**

---

**Q16. Rotating a matrix 90° clockwise is equivalent to:** a) Transpose + Reverse rows
b) Transpose + Reverse columns
c) Only transpose
d) Only reverse rows
**Answer: (a) Transpose + Reverse rows**

---

**Q17. Which algorithm is more efficient for large matrix multiplication?** a) Naive method
b) Strassen's Algorithm
c) Gaussian Elimination
d) QuickSort
**Answer: (b) Strassen's Algorithm**

---

**Q18. In Strassen's Algorithm, time complexity reduces to:** a) $O(n^2)$
b) $O(n^{2.81})$
c) $O(n^3)$
d) $O(\log n)$
**Answer: (b) $O(n^{2.81})$**

---

**Q19. The determinant of a matrix is defined only for:** a) Rectangular matrices
b) Square matrices
c) Identity matrices
d) Symmetric matrices **Answer: (b) Square matrices**

---

**Q20. The inverse of a matrix exists only if:** a) Matrix is symmetric
b) Determinant ≠ 0
c) Diagonal elements are non-zero
d) Matrix is diagonal
**Answer: (b) Determinant ≠ 0**

---

## Sparse Matrices

**Q21. A sparse matrix is one where:** a) All elements are 0
b) Non-zero elements < 50% of total
c) Non-zero elements > 50% of total
d) All diagonal elements are 1
**Answer: (b) Non-zero elements < 50% of total**

---

**Q22. Common representation of a sparse matrix is:** a) Linked list
b) 3-tuple form (row, col, value)
c) Hash table
d) Array of size $n^2$
**Answer: (b) 3-tuple form (row, col, value)**

---

**Q23. Which of the following is NOT an advantage of sparse representation?** a) Saves memory
b) Faster traversal of non-zeros
c) Easier arithmetic operations
d) Always faster than dense representation **Answer: (c) Easier arithmetic operations**

---

**Q24. What is the maximum number of non-zero elements a sparse matrix can have?** a) < n × m
b) = n × m
c) n
d) Undefined **Answer: (b) = n × m**

---

**Q25. Time complexity of adding two sparse matrices (triplet form):** a) $O(n^2)$

b) O(n)
c) O(number of non-zero elements)
d) O(1)
**Answer: (c) O(number of non-zero elements)**

---

**Q26. If a sparse matrix has 1000 rows × 1000 columns with only 1000 non-zero elements, memory saved using sparse representation is approximately:** a) Very small
b) Significant
c) No difference
d) More costly
**Answer: (b) Significant**

---

**Q27. Which storage is most efficient for sparse matrices in numerical computations?** a) Dense array
b) Compressed Sparse Row (CSR)
c) Hash table
d) Linked list
**Answer: (b) Compressed Sparse Row (CSR)**

---

**Q28. In triplet representation, the first row usually stores:** a) Matrix order and count of non-zero elements
b) Only rows
c) Only columns
d) Only diagonal
**Answer: (a) Matrix order and count of non-zero elements**

---

**Q29. Which of the following applications commonly uses sparse matrices?** a) Graphs (Adjacency Matrix)
b) Image compression
c) Machine learning (large datasets)
d) All of the above
**Answer: (d) All of the above**

---

**Q30. Which of the following is true about sparse matrices?** a) They always store fewer elements
b) They only store diagonal elements
c) They reduce memory usage when most entries are 0
d) They cannot be used in graph problems
**Answer: (c) They reduce memory usage when most entries are 0**

# SECTION C:

## Transpose of Matrix

```
import java.util.Scanner;


public class TransposeMatrix {
public          static          void
main(String[] args) {
      Scanner sc = new Scanner(System.in);


      // Input matrix dimensions
System.out.print("Enter rows: ");
int n = sc.nextInt();
      System.out.print("Enter columns:
");       int m = sc.nextInt();


      int[][] matrix = new int[n][m];
      System.out.println("Enter elements of matrix:");
      for (int i = 0; i < n; i++) {
for (int j = 0; j < m; j++) {
matrix[i][j] = sc.nextInt();
         }
      }


      // Print transpose
      System.out.println("Transpose of Matrix:");
```

```
        for (int i = 0; i < m; i++)
{           for (int j = 0; j < n;
j++) {
              System.out.print(matrix[j][i] + " ");
           }
           System.out.println();
        }
    }
}
```

## Rotate Matrix (90° Clockwise)

```
import java.util.Scanner;

public class RotateMatrix {
public static void main(String[]
args) {
        Scanner sc = new Scanner(System.in);

        // Input size of square matrix
        System.out.print("Enter size of square matrix
(n): ");  int n = sc.nextInt();

        int[][] matrix = new int[n][n];
        System.out.println("Enter elements of matrix:");
        for (int i = 0; i < n; i++)
{           for (int j = 0; j < n;
j++) {              matrix[i][j]
= sc.nextInt();
```

```
        }
    }

    // Step 1: Transpose
for (int i = 0; i < n; i++) {
for (int j = i; j < n; j++) {
int temp = matrix[i][j];
matrix[i][j] = matrix[j][i];
matrix[j][i] = temp;
        }
    }

    // Step 2: Reverse each row
    for (int i = 0; i < n; i++) {
int left = 0, right = n - 1;
while (left < right) {              int
temp = matrix[i][left];
matrix[i][left] =
matrix[i][right]; matrix[i][right]
= temp;
left++;             right--;
        }
    }

    // Print rotated matrix
    System.out.println("Matrix after 90° Clockwise Rotation:");
    for (int i = 0; i < n; i++)
{          for (int j = 0; j < n;
j++) {
```

```
            System.out.print(matrix[i][j] + " ");
        }
        System.out.println();
    }
  }
}
```