**Set A**

* **Python**

def decryptMatrix(matrix):

rows, cols = np.shape(matrix)

result = np.zeros((cols, rows), dtype=int)

for r in range(rows):

for c in range(cols):

result[c][rows - r - 1] = matrix[r][c]

return result

matrix = np.array([

[4,5,2],

[21,67,3],

])

print(decryptMatrix(matrix))

print("---------------------------------")

matrix2 = np.array([

[4,5,12],

[21,67,3],

[1, 42, 55]

])

print(decryptMatrix(matrix2))

* **Java**

public static int[][] decryptMatrix(int[][] matrix) {

int rows = matrix.length;

int cols = matrix[0].length;

int[][] result = new int[cols][rows];

for (int r = 0; r < rows; r++) {

for (int c = 0; c < cols; c++) {

result[c][rows - r - 1] = matrix[r][c];

}

}

return result;

}

**Set B**

* **Python**

def decodeMessage(matrix):

rows, cols = np.shape(matrix)

result = np.zeros((cols, rows), dtype=int)

for r in range(rows):

for c in range(cols):

result[c][r] = matrix[r][c]

return result

matrix = np.array([

[4,5,2],

[21,67,3],

])

print(decodeMessage(matrix))

print("---------------------------------")

matrix2 = np.array([

[4,5,12],

[21,67,3],

[1, 42, 55]

])

print(decodeMessage(matrix2))

* **Java**

public static int[][] decodeMessage(int[][] matrix) {

int rows = matrix.length;

int cols = matrix[0].length;

// Create a new matrix to store the result with swapped dimensions

int[][] result = new int[cols][rows];

// Transpose the matrix

for (int r = 0; r < rows; r++) {

for (int c = 0; c < cols; c++) {

result[c][r] = matrix[r][c];

}

}

return result;

}

# **RUBRIC**

| Finding array shape (row and column) | 3 |
| --- | --- |
| Iterate through each row and column | 2 + 2 |
| LHS of the value assignment | 3 |
| RHS of the value assignment | 3 |
| Returning the modified array | 2 |
| **Total** | **15** |