Lab 9

Name: Yuvraj Singh
Roll No: 23072021
M.Tech CSE

Problem: Write a program that takes p (a prime) as input, an $m \times n$ matrix A, and Z_p, and outputs its Reduced Row Echelon (RRE) form.

Solution:

The Reduced Row Echelon form (RRE) of a matrix is a specific form achieved through a series of row operations. In the RRE form:

1. Each leading entry (pivot) of a row is 1.
2. The leading entry is the only non-zero entry in its column.
3. Rows with all zero entries, if any, are at the bottom of the matrix.
4. Each leading 1 is to the right of the leading 1 of the row above it.

Example: Consider the following matrix:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

To reduce this matrix to its Reduced Row Echelon form (RRE), we'll perform row operations to transform it. Here's the step-by-step process:

1. We'll begin by making the leading entry (pivot) of the first column 1 and eliminating other entries below it.
2. Subtract the first row from the third row:
$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 0 & 6 & 5 \end{pmatrix}$$
3. Subtracting the first row from the second row yields:
$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 6 & 2 \\ 0 & 6 & 5 \end{pmatrix}$$
4. To make the leading entry of the third row 1, we'll divide the entire row by 3 (means multipy by $3^{-1} = 5$:
$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 6 & 2 \\ 0 & 0 & 3 \end{pmatrix}$$
5. Now, to make the leading entry of the second row 1, we'll divide the entire row by 6 (means multipy by $6^{-1} = 6$:
$$A = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{pmatrix}$$

6. This is the Reduced Row Echelon form (RRE) of the matrix. Each leading entry (pivot) is 1, and each leading entry is the only non-zero entry in its column.

Implementation in C++

```cpp
#include <iostream>
#include <vector>
using namespace std;

class Matrix {
private:
    vector<vector<int>> data;
    int p; // Prime
    // Function to perform modulo operation
    int mod(int a, int b) {
        int result = a % b;
        if (result < 0) result += b;
        return result;
    }
    // Function to perform subtraction in Zp
    int subtract(int a, int b) {
        return mod(a - b, p);
    }
    int multiply(int a, int b) {
        return mod(a * b, p);
    }
    // Function to calculate modular exponentiation (a^b mod p)
    int modExp(int base, int exponent) {
        long long result = 1;
        while (exponent > 0) {
            if (exponent & 1)
                result = (result * base) % p;
            exponent >>= 1;
            base = (base * base) % p;
        }
        return static_cast<int>(result);;
    }
    // Function to calculate modular inverse (a^(-1) mod p)
    int modInverse(int a) {
        return modExp(a, p - 2);
    }
public:
    Matrix(int m, int n, int prime) : data(m, vector<int>(n)), p(prime) {}

    void setElement(int row, int col, int value) {
        data[row][col] = value % p; // Ensure element belongs to Z_p
    }

    void printMatrix() const {
        for (const auto& row : data) {
            for (int elem : row)
                cout << elem << " ";
            cout << endl;
        }
    }
```

```cpp

    void rowReduce() {
        for (int col = 0; col < data.size(); ++col) {
            // Make the pivot element 1
            int pivot = data[col][col];
            int inv = modInverse(pivot);
            for (int j = col; j < data.size() + 1; ++j)
                data[col][j] = multiply(data[col][j], inv);

            for (int row = col+1; row < data.size(); ++row) {
                int factor = data[row][col];
                for (int j = col; j < data.size() + 1; ++j) {
                    int prod = multiply(factor, data[col][j]);
                    data[row][j] = subtract(data[row][j], prod);
                }
            }
        }
    }
};

int main() {
    int p, m, n;;
    cout << "Enter the prime number p: "; cin >> p;
    cout << "Enter the dimensions of the matrix (m x n): "; cin >> m >> n;

    Matrix A(m, n, p); // Create a matrix with prime modulus p
    cout << "Enter the elements of the matrix A (mod " << p << "):\n";
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
            int value;
            cin >> value;
            A.setElement(i, j, value);
        }
    }
    A.rowReduce(); // Perform row reduction
    cout << "\nRow Reduced Echelon Form of matrix A:\n";
    A.printMatrix(); // Output the row-reduced echelon form
    return 0;
}
```

Time Complexity: $O(m \times n)$

Output:

```
Enter the prime number p: 7
Enter the dimensions of the matrix (m x n): 3 3
Enter the elements of the matrix A (mod 7):
1 2 3
2 3 1
1 1 1

Row Reduced Echelon Form of matrix A:
1 2 3
0 1 5
0 0 1
```