INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
DEPARTMENT OF MATHEMATICS AND STATISTICS

Class Test - 9 **MA517M-Basic Programming Laboratory** 13 October 2025

Name **Roll No.: MA25M**

Download the Vector Class and complete the missing code
(`https://facweb.iittp.ac.in/~pmariappan/Courses/MA517M/VectorClassTest9.cpp`)

```
#include <iostream>
#include <cmath>
#include <fstream>
using namespace std;

class Vector {
    private:
        int length;
        double *values;
    public:
        // Constructor: creates a vector of length N, initialized with 'val'
        Vector(int N = 0, double val = 0.0) : length(N) {
            if (N > 0) {
                values = new double[N];
                for (int i = 0; i < N; i++) values[i] = val;
            } else values = nullptr;
        }

        Vector(const Vector &v) : length(v.length) {   // Copy constructor
            values = new double[length];
            for (int i = 0; i < length; i++)
                values[i] = v.values[i];
        }

        ~Vector() { delete[] values; }     // Destructor

        Vector& operator=(const Vector &v) {    // Assignment operator
            if (this != &v) {
                delete[] values;
                length = v.length;
                values = new double[length];
                for (int i = 0; i < length; i++)
                    values[i] = v.values[i];
            }
            return *this;
        }
        // Vector addition (friend function)
        friend Vector operator+(const Vector &x, const Vector &y) {
            Vector result(x.length);
            for (int i = 0; i < x.length; i++)
                result.values[i] = x.values[i] + y.values[i];
```

```
            return result;      }

        friend Vector operator-(const Vector &x, const Vector &y) {
        // Vector subtraction Fill The code}

        friend Vector operator*(const Vector &x, double c) {
         // Scalar multiplication (c * x) Fill the code }

        friend Vector operator*(double c, const Vector &x) {
            // Scalar multiplication (x * c) Fill the code

        }

        double& operator[](int i) const { return values[i]; } // Indexing operator


        double norm2() {
               // Euclidean norm (L2) Fill the code
        }

        double infnorm() {
            // Infinity norm Fill the code
        }

        int size() const { return length; }    // Return size

        void print() {// Print vector
            cout << "[ ";
            for (int i = 0; i < length; i++)
                cout << values[i] << " ";
            cout << "]" << endl;
        }
};
int main() {
    Vector a(3, 2.0), b(3, 1.0);
    Vector c = a + b;
    Vector d = a - b;
    Vector e = 2.5 * a;

    cout << "Vector a: "; a.print();
    cout << "Vector b: "; b.print();
    cout << "a + b = "; c.print();
    cout << "a - b = "; d.print();
    cout << "2.5 * a = "; e.print();

    cout << "Norm of a = " << a.norm() << endl;
    cout << "Infinity norm of e = " << e.infnorm() << endl;

    return 0;
}
```

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI
DEPARTMENT OF MATHEMATICS AND STATISTICS

Class Test - 9 **MA517M-Basic Programming Laboratory** 13 October 2025

Name **Roll No.: MA25M**

Download the Vector Class and complete the missing code
(`https://facweb.iittp.ac.in/~pmariappan/Courses/MA517M/MatrixClassTest9.cpp`)

```
#include <iostream>
#include <cmath>
#include <sstream>
#include <stdexcept>
using namespace std;

class Matrix {
    private:
        double *values;
        int row, col;
    public:
        Matrix(int M = 0, int N = 0, double x = 0.0) : row(M), col(N) { // Constructor
            if (M > 0 && N > 0) {
                values = new double[row * col];
                for (int i = 0; i < row * col; i++)
                    values[i] = x;
            } else
                values = nullptr;
        }
        Matrix(double *val, int r, int c) : row(r), col(c) { // Constructor from array
            values = new double[row * col];
            for (int i = 0; i < row * col; i++)
                values[i] = val[i];
        }

        Matrix(const Matrix &A) : row(A.row), col(A.col) { // Copy constructor
            values = new double[row * col];
            for (int i = 0; i < row * col; i++)
                values[i] = A.values[i];
        }
        ~Matrix() { delete[] values; } // Destructor
        Matrix& operator=(const Matrix &A) { // Assignment operator
            if (this != &A) {
                delete[] values;
                row = A.row;
                col = A.col;
                values = new double[row * col];
                for (int i = 0; i < row * col; i++)
                    values[i] = A.values[i];
            }
            return *this;      }
```

```
        double& operator[](int i) const { // Indexing operator (access as 1D array)
            if (i < 0 || i >= row * col) {
                throw out_of_range("Matrix index out of range");
            }
            return values[i];      }

        friend Matrix operator+(const Matrix &A, const Matrix &B) { // Matrix addition
            if (A.row != B.row || A.col != B.col)
                throw invalid_argument("Matrix dimensions must match for addition");
            Matrix C(A.row, A.col);
            for (int i = 0; i < A.row * A.col; i++)
                C.values[i] = A.values[i] + B.values[i];
            return C;
        }

        friend Matrix operator-(const Matrix &A, const Matrix &B) { // Matrix subtraction
        //fill the missing code}

        friend Matrix operator*(double c, const Matrix &A) {// Scalar multiplication (c*A)
        //fill the missing code }

        double norm1() const {// L1 norm (maximum absolute column sum)
        //fill the missing code}

        double norm2() const { // L2 norm (Frobenius norm)
        //fill the missing code }

        void print() const {  // Print the matrix
            for (int i = 0; i < row; i++) {
                cout << "[ ";
                for (int j = 0; j < col; j++)
                    cout << values[i * col + j] << " ";
                cout << "]" << endl; }}
        int nrows() const { return row; }
        int ncols() const { return col; }
};
int main() {
    double arr[] = {1,2,3,4,5,6};
    Matrix A(arr, 2, 3);
    Matrix B(2, 3, 1.0);
    Matrix C = A + B; C.print();
    C = A - B; C.print();
    C = 2.5*A; C.print();
    Matrix A(2,3);
    A[0]=1; A[1]=-2; A[2]=3; A[3]=-4; A[4]=5; A[5]=-6;

    cout << "L1 norm of A = " << A.norm1() << endl;
    cout << "L2 norm of A = " << A.norm2() << endl;
    return 0; }
```