Laborator 1

16. Scrieți funcții pentru implemetarea operațiilor specifice pe matrice de numere reale cu m linii și n coloane: suma, diferența și produsul al două matrice, produsul dintre o matrice și un scalar real, transpusa unei matrice, norme matriceale specifice[1], citirea de la tastatură a componentelor unei matrice, afișarea componentelor matricei. Pentru cazul particular al unei matrice patratice de ordin n, să se testeze dacă aceasta satisface criteriul de dominanță pe linii[2] sau pe coloane[3]. Se vor folosi tablouri bidimensionale alocate static.

```cpp
#include <iostream>
#include <cmath>

using namespace std;
#define MAX_SZ 8

// const is optional, through recommended
double norm1(const int A[MAX_SZ][MAX_SZ], int m, int n) {
    double max = -1;
    for (int j = 0; j < m; ++j) {
        double x = 0;
        for (int i = 0; i < n; ++i) {
            x += abs(A[i][j]);
        }
        if (x > max) {
            max = x;
        }
    }
    return max;
}

double norm2(const int A[MAX_SZ][MAX_SZ], int m, int n) {
    double max = -1;
    for (int j = 0; j < n; ++j) {
        double x = 0;
        for (int i = 0; i < m; ++i)
            x += abs(A[j][i]);
        if (x > max) max = x;
    }
    return max;
}

double normF(const int A[MAX_SZ][MAX_SZ], int m, int n) {
    double res = 0;
    for (int i = 0; i < m; ++i)
```

[1] Dacă $A \in \mathcal{M}_{m\times n}(\mathbb{R})$, atunci $||A||_1 = \max\limits_{1\le j\le n} \sum\limits_{i=1}^{m} |a_{ij}|$, $||A||_\infty = \max\limits_{1\le i\le m} \sum\limits_{j=1}^{n} |a_{ij}|$, $||A||_F = \sqrt{\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} a_{ij}^2}$.

[2] $A \in \mathcal{M}_n(\mathbb{R})$ este strict diagonal dominantă pe linii dacă $|a_{ii}| > \sum\limits_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|$, pentru orice $i = 1, ..., n$.

[3] $A \in \mathcal{M}_n(\mathbb{R})$ este strict diagonal dominantă pe colonane dacă $|a_{jj}| > \sum\limits_{\substack{i=1\\ i\neq j}}^{n} |a_{ij}|$, pentru orice $j = 1, ..., n$.

```
        for (int j = 0; j < n; ++j)
            res += A[i][j] * A[i][j];

    return sqrt(res);
}
int do_something_undefined() {
    return 1 / 0;
}

bool isStrictlyRowDiagonallyDominant(const int A[MAX_SZ][MAX_SZ],
                                     int m, int n) {
    if (m != n) {
        cout << "launching nuclear missile because"
             << "matrix is not square\n";
        do_something_undefined();
    }

    for (int i = 0; i < m; ++i) {
        int val = std::abs(A[i][i]);
        int sum = -val;
        for (int j = 0; j < m; ++j)
            sum += abs(A[i][j]);
        if (sum >= val) return false;
    }
    return true;
}
//implement this, natürlich
void transpose(const int A[MAX_SZ][MAX_SZ], int res[MAX_SZ][MAX_SZ],
               int m, int n);

//not the most efficient, but good enough for demonstration purposes
bool isStrictlyColDiagonallyDominant(const int A[MAX_SZ][MAX_SZ],
                                     int m, int n) {
    int res[MAX_SZ][MAX_SZ];
    transpose(A, res, m, n);
    return isStrictlyRowDiagonallyDominant(res, n, m);
}
```