```cpp
//
//   There is a simple directed graph G with N vertices, numbered 1,2,…,N.
//   For each i and j (1 ⩽ i,j ⩽ N), you are given an integer a[i][j]
//   that represents whether there is a directed edge from Vertex i to j if a[i][j] is set.
//   Find the number of different directed paths of length K in G, modulo 10^9 + 7
//   We will also count a path that traverses the same edge multiple times.
//
//   Constraints:
//     N ⩽ 50
//     K ⩽ 10^18
//
//   Time Complexity: O(N^3 * log(K))
//

#include <bits/stdc++.h>
#define MX vector<vector<T>>
#define MOD 1000000007
#define ll long long

using namespace std;

template<typename T>
MX mult(MX a, MX b) {
    MX result(a.size(), vector<T>(a.size(), 0));
    for (int i = 0; i < a.size(); i++) {
        for (int j = 0; j < a.size(); j++) {
            for (int k = 0; k < a.size(); k++) {
                result[i][j] = (result[i][j] + (a[i][k] * b[k][j]) % MOD) % MOD;
            }
        }
    }

    return result;
}

template<typename T>
MX binary(MX a, ll n) {
    MX result(a.size(), vector<T>(a.size(), 0));
    for (int i = 0; i < a.size(); i++) result[i][i] = 1;

    while (n > 0) {
        if (n % 2 == 1) {
            result = mult(result, a);
            n--;
        }

        a = mult(a, a);
        n /= 2;
    }

    return result;
}

int main() {
    ll n, k;
    cin >> n >> k;

    vector<vector<long long>> mat(n, vector<long long>(n, 0));

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> mat[i][j];
        }
    }

    auto result = binary(mat, k);

    long long sum = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            sum = (sum + result[i][j]) % MOD;
        }
    }

    cout << sum << endl;
    return 0;
}
```