Lego Blocks

```cpp
#include <bits/stdc++.h>

using namespace std;

string ltrim(const string &);
string rtrim(const string &);
vector<string> split(const string &);

/*
 * Complete the 'legoBlocks' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 *  1. INTEGER n
 *  2. INTEGER m
 */

const int MOD = 1000000007;

long long modPow(long long base, long long exp) {
    long long res = 1;
    while (exp > 0) {
        if (exp & 1) res = (res * base) % MOD;
        base = (base * base) % MOD;
        exp >>= 1;
    }
    return res;
}

int legoBlocks(int n, int m) {
    vector<long long> row(m + 1, 0);
    row[0] = 1;
    for (int i = 1; i <= m; i++) {
        for (int b = 1; b <= 4; b++) {
            if (i - b >= 0) row[i] = (row[i] + row[i - b]) % MOD;
        }
    }

    vector<long long> total(m + 1, 0);
    for (int i = 1; i <= m; i++) {
        total[i] = modPow(row[i], n);
    }
    vector<long long> solid(m + 1, 0);
    for (int i = 1; i <= m; i++) {
        solid[i] = total[i];
        for (int k = 1; k < i; k++) {
            solid[i] = (solid[i] - (solid[k] * total[i - k]) %
MOD + MOD) % MOD;
        }
    }

    return (int)solid[m];
}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    string t_temp;
    getline(cin, t_temp);

    int t = stoi(ltrim(rtrim(t_temp)));

    for (int t_itr = 0; t_itr < t; t_itr++) {
        string first_multiple_input_temp;
        getline(cin, first_multiple_input_temp);

        vector<string> first_multiple_input =
split(rtrim(first_multiple_input_temp));

        int n = stoi(first_multiple_input[0]);

        int m = stoi(first_multiple_input[1]);

        int result = legoBlocks(n, m);

        fout << result << "\n";
    }

    fout.close();

    return 0;
}

string ltrim(const string &str) {
    string s(str);

    s.erase(
        s.begin(),
        find_if(s.begin(), s.end(), not1(ptr_fun<int,
int>(isspace)))
    );

    return s;
}

string rtrim(const string &str) {
    string s(str);

    s.erase(
        find_if(s.rbegin(), s.rend(), not1(ptr_fun<int,
int>(isspace))).base(),
        s.end()
    );

    return s;
}

vector<string> split(const string &str) {
    vector<string> tokens;

    string::size_type start = 0;
    string::size_type end = 0;

    while ((end = str.find(" ", start)) != string::npos) {
        tokens.push_back(str.substr(start, end - start));

        start = end + 1;
    }

    tokens.push_back(str.substr(start));

    return tokens;
}
```