Roads and Libraries

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

using namespace std;

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

/*
 * Complete the 'roadsAndLibraries' function below.
 *
 * The function is expected to return a LONG_INTEGER.
 * The function accepts following parameters:
 *  1. INTEGER n
 *  2. INTEGER c_lib
 *  3. INTEGER c_road
 *  4. 2D_INTEGER_ARRAY cities
 */

long roadsAndLibraries(int n, int c_lib, int c_road,
vector<vector<int>> cities) {
    if(c_lib<=c_road) return (long)n*c_lib;
    vector<vector<int>> adj(n+1);
    for(auto &e:cities){
        adj[e[0]].push_back(e[1]);
        adj[e[1]].push_back(e[0]);
    }
    vector<int> vis(n+1,0);
    long cost=0;
    for(int i=1;i<=n;i++){
        if(!vis[i]){
            stack<int> st;
            st.push(i);
            vis[i]=1;
            long cnt=0;
            while(!st.empty()){
                int u=st.top();st.pop();
                cnt++;
                for(int v:adj[u]) if(!vis[v]){
                    vis[v]=1;
                    st.push(v);
                }
            }
```

```
            cost+=c_lib+(cnt-1)*c_road;
        }
    }
    return cost;
}

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

    string q_temp;
    getline(cin, q_temp);

    int q = stoi(ltrim(rtrim(q_temp)));

    for (int q_itr = 0; q_itr < q; q_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 c_lib = stoi(first_multiple_input[2]);

        int c_road = stoi(first_multiple_input[3]);

        vector<vector<int>> cities(m);

        for (int i = 0; i < m; i++) {
            cities[i].resize(2);

            string cities_row_temp_temp;
            getline(cin, cities_row_temp_temp);

            vector<string> cities_row_temp =
split(rtrim(cities_row_temp_temp));

            for (int j = 0; j < 2; j++) {
                int cities_row_item = stoi(cities_row_temp[j]);

                cities[i][j] = cities_row_item;
            }
```

```
        }

        long result = roadsAndLibraries(n, c_lib, c_road,
cities);

        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;
}
```