```cpp
#include <iostream>
#include <random>
#include <iomanip>
#include <vector>
#include <fstream>
#include <sstream>
using namespace std;
typedef vector<vector<float>> mgraph;
struct ledge{
	int v;
	float w;
};
typedef vector<vector<ledge>>lgraph;
lgraph buildLgraph(int n) {
	// creat n rows
	lgraph l(n);
	for (int i = 0; i < n; i++) {
		l[i] = vector<ledge>(0, { 0 });
	}
	return l;
}

void printLgraph(lgraph l) {
	for (int i = 0; i < l.size(); i++) {
		cout << "[" << i << "]: ";
		for (int j = 0; j < l[i].size(); j++) {
			cout << "[" << l[i][j].v << " " << l[i][j].w << "]" << " ";
		}
		cout << endl;
	}
}

void addLedge(lgraph& l, int u, ledge e) {
	if (u < 0 || u > l.size() || e.v < 0 || e.v > l.size()) {
		return;
	}
	if (u == e.v) {
		return;
	}
	l[u].push_back(e);
	l[e.v].push_back({ u, e.w });
}

mgraph buildMgraph(int numOfvertices) {
	mgraph g(numOfvertices, vector<float>(numOfvertices, 0));
	return g;
}

void printMgraph(mgraph g) {
	for (int i = 0; i < g.size(); i++) {
		for (int j = 0; j < g[i].size(); j++) {
			cout << fixed << setprecision(1) << g[i][j] << " ";
		}
		cout << endl;
	}
}

void addMedge(mgraph& g, int u, int v, float w) {
	if (u < 0 || u > g.size() || v < 0 || v > g.size()) {
		return;
	}
	g[u][v] = w;
}
```

```
vector <string> getWord(string s) {
	stringstream ss(s);
	string word;
	vector <string> v;

	while (ss >> word) {
		v.push_back(word);
	}
	return v;
}

mgraph createMgraph() {
	ifstream f("graph.txt");
	string s;

	getline(f, s); // first containing list of vertices

	vector <string> vertices = getWord(s);
	mgraph g = buildMgraph(vertices.size());

	while (getline(f, s)) {
		vector <string> v = getWord(s);
		auto a = find(vertices.begin(), vertices.end(), v[0]);
		auto b = find(vertices.begin(), vertices.end(), v[1]);
		addMedge(g, distance(vertices.begin(), a),
distance(vertices.begin(), b), 1);
	}
	f.close();
	return g;
}

lgraph createLgraph() {
	ifstream f("graph.txt");
	string s;

	getline(f, s); // first containing list of vertices

	vector <string> vertices = getWord(s);
	lgraph l = buildLgraph(vertices.size());

	// Get each pair on input
	while (getline(f, s)) {
		vector <string> v = getWord(s);
		auto a = find(vertices.begin(), vertices.end(), v[0]);
		auto b = find(vertices.begin(), vertices.end(), v[1]);
		addLedge(l, distance(vertices.begin(), a), {
(int)distance(vertices.begin(),b), 1.0});
	}
	f.close();
	return l;
}

int main() {
	cout << "\nMGraph: ";
	cout << endl;
	mgraph g;
	g = createMgraph();
	printMgraph(g);

	cout << "\nLgraph: ";
	cout << endl;
	lgraph l;
```

```
	l = createLgraph();
	printLgraph(l);
	return 0;
}



#include <iostream>
#include <random>
#include <iomanip>
#include <vector>
#include <fstream>
#include <sstream>
using namespace std;
typedef vector<vector<float>> mgraph;
struct ledge{
	int v;
	float w;
};
typedef vector<vector<ledge>>lgraph;
lgraph buildLgraph(int n) {
	// creat n rows
	lgraph l(n);
	for (int i = 0; i < n; i++) {
		l[i] = vector<ledge>(0, { 0 });
	}
	return l;
}

void printLgraph(lgraph l) {
	for (int i = 0; i < l.size(); i++) {
		cout << "[" << i << "]: ";
		for (int j = 0; j < l[i].size(); j++) {
			cout << "[" << l[i][j].v << " " << l[i][j].w << "]" << " ";
		}
		cout << endl;
	}
}

void addLedge(lgraph& l, int u, ledge e) {
	if (u < 0 || u > l.size() || e.v < 0 || e.v > l.size()) {
		return;
	}
	if (u == e.v) {
		return;
	}
	l[u].push_back(e);
	l[e.v].push_back({ u, e.w });
}

mgraph buildMgraph(int numOfvertices) {
	mgraph g(numOfvertices, vector<float>(numOfvertices, 0));
	return g;
}

void printMgraph(mgraph g) {
	for (int i = 0; i < g.size(); i++) {
		for (int j = 0; j < g[i].size(); j++) {
			cout << fixed << setprecision(1) << g[i][j] << " ";
		}
		cout << endl;
	}
}
```

```
void addMedge(mgraph& g, int u, int v, float w) {
	if (u < 0 || u > g.size() || v < 0 || v > g.size()) {
		return;
	}
	g[u][v] = w;
}

vector <string> getWord(string s) {
	stringstream ss(s);
	string word;
	vector <string> v;

	while (ss >> word) {
		v.push_back(word);
	}
	return v;
}

mgraph createMgraph() {
	ifstream f("graph.txt");
	string s;

	getline(f, s); // first containing list of vertices

	vector <string> vertices = getWord(s);
	mgraph g = buildMgraph(vertices.size());

	while (getline(f, s)) {
		vector <string> v = getWord(s);
		auto a = find(vertices.begin(), vertices.end(), v[0]);
		auto b = find(vertices.begin(), vertices.end(), v[1]);
		addMedge(g, distance(vertices.begin(), a),
distance(vertices.begin(), b), 1);
	}
	f.close();
	return g;
}

lgraph createLgraph() {
	ifstream f("graph.txt");
	string s;

	getline(f, s); // first containing list of vertices

	vector <string> vertices = getWord(s);
	lgraph l = buildLgraph(vertices.size());

	// Get each pair on input
	while (getline(f, s)) {
		vector <string> v = getWord(s);
		auto a = find(vertices.begin(), vertices.end(), v[0]);
		auto b = find(vertices.begin(), vertices.end(), v[1]);
		addLedge(l, distance(vertices.begin(), a), {
(int)distance(vertices.begin(),b), 1.0});
	}
	f.close();
	return l;
}

int main() {
	cout << "\nMGraph: ";
	cout << endl;
```

```
        mgraph g;
        g = createMgraph();
        printMgraph(g);

        cout << "\nLgraph: ";
        cout << endl;
        lgraph l;
        l = createLgraph();
        printLgraph(l);
        return 0;
}
```