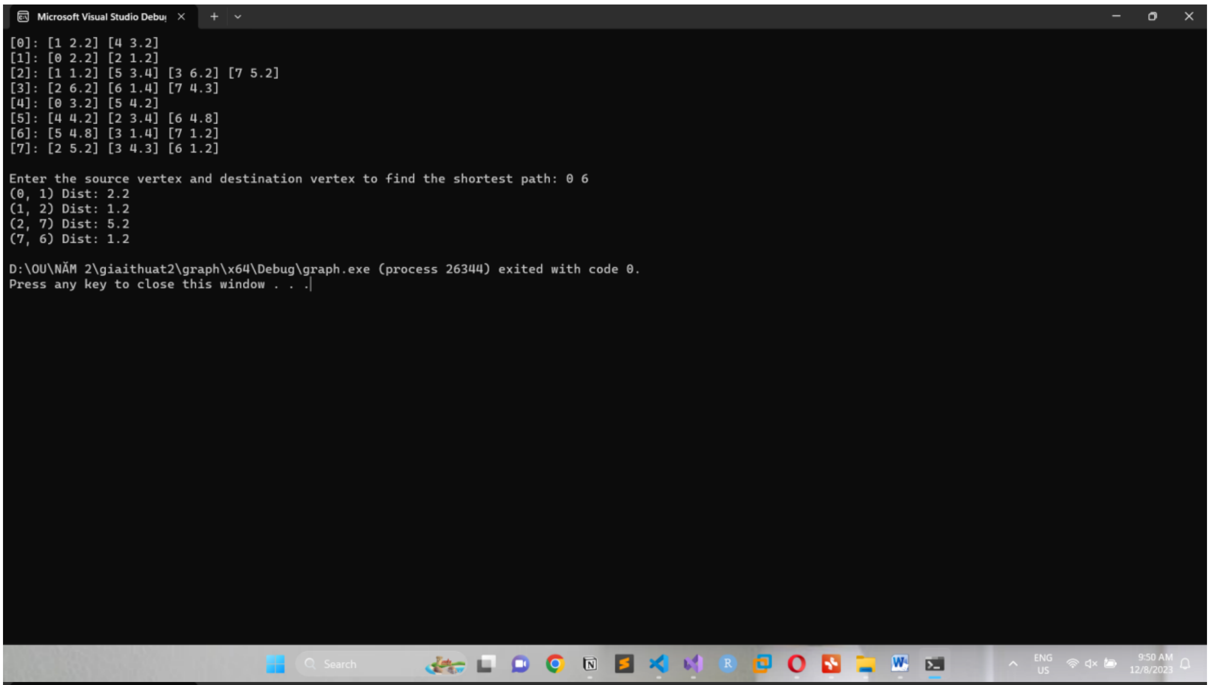
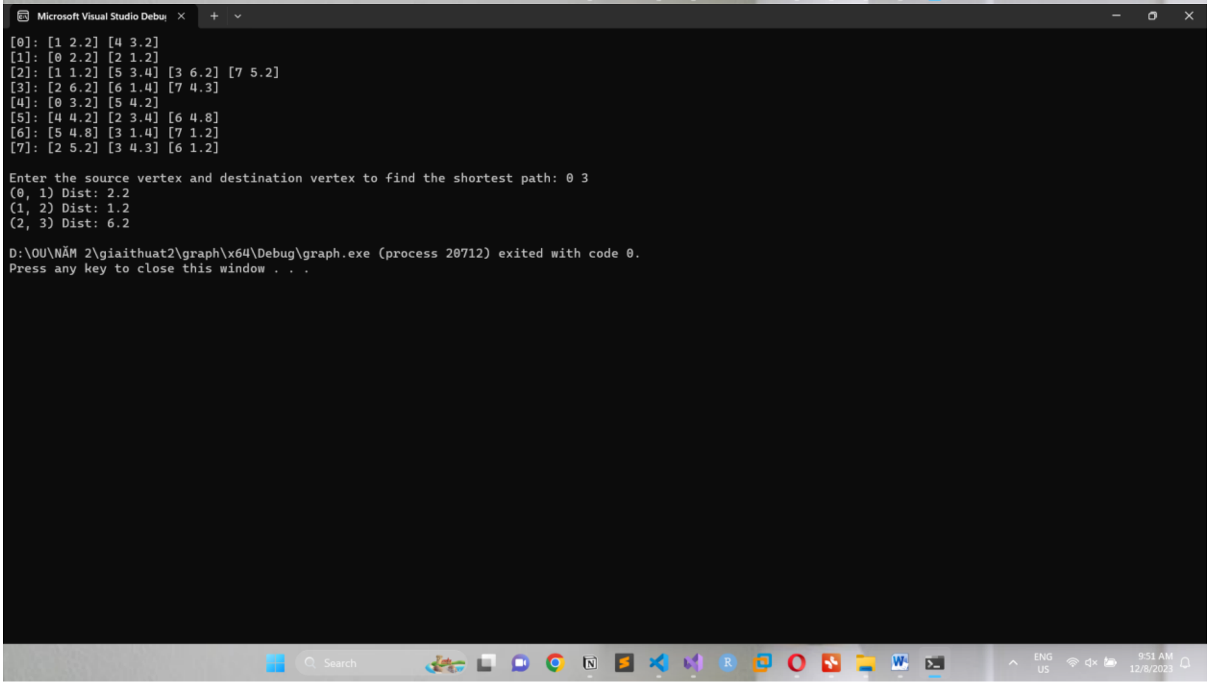
```
void printPath(vector <int> prev, vector <float> dist, int source, int
destination) {
    if (destination < 0) {
        return;
    }
    printPath(prev, dist, source, prev[destination]);
    if (destination != -1 && prev[destination] != -1) {
        cout << "(" << prev[destination] << ", " << destination << ") ";
        cout << "Dist: " << dist[destination] - dist[prev[destination]] <<
endl;
    }
}

vector <float> bellmanFord(lgraph l, int  u, vector <int>& prev) {
    // set intial distance form the source to v as infinity
    vector <float> dist(l.size(), INT_MAX);
    dist[u] = 0;
    // all vertices
    for (int i = 1; i < l.size(); i++) {
        // for each vertex, get all edges
        for (int j = 0; j < l.size(); j++) {
            for (int k = 0; k < l[j].size(); k++) {
                int v = l[j][k].v;
                float w = l[j][k].w;
                if (dist[j] + w < dist[v]) {
                    dist[v] = dist[j] + w;
                    prev[v] = j;
                }
            }
        }
    }
    return dist;
}



int main() {
    lgraph l;
    l = createLgraph();

    printLgraph(l);
    vector <int> prev(l.size());
    for (int i = 0; i < prev.size(); i++) {
        prev[i] = -1;
    }

    int source, destination;
    do {
        cout << "\nEnter the source vertex and destination vertex to find
the shortest path: ";
        cin >> source >> destination;
        if (source < 0 || source > l.size() || destination < 0 ||
destination > l.size()) {
            cout << "\nInvalid value. Please re-enter";
        }
    } while (source < 0 || source > l.size() || destination < 0 || destination
> l.size());

    vector <float> bellman = bellmanFord(l, source, prev);
    printPath(prev,bellman,source,destination);
    return 0;
}
```



```
[0]: [1 2.2] [4 3.2]
[1]: [0 2.2] [2 1.2]
[2]: [1 1.2] [5 3.4] [3 6.2] [7 5.2]
[3]: [2 6.2] [6 1.4] [7 4.3]
[4]: [0 3.2] [5 4.2]
[5]: [4 4.2] [2 3.4] [6 4.8]
[6]: [5 4.8] [3 1.4] [7 1.2]
[7]: [2 5.2] [3 4.3] [6 1.2]

Enter the source vertex and destination vertex to find the shortest path: 0 6
(0, 1) Dist: 2.2
(1, 2) Dist: 1.2
(2, 7) Dist: 5.2
(7, 6) Dist: 1.2

D:\OU\NĂM 2\giaithuat2\graph\x64\Debug\graph.exe (process 26344) exited with code 0.
Press any key to close this window . . .
```



```
[0]: [1 2.2] [4 3.2]
[1]: [0 2.2] [2 1.2]
[2]: [1 1.2] [5 3.4] [3 6.2] [7 5.2]
[3]: [2 6.2] [6 1.4] [7 4.3]
[4]: [0 3.2] [5 4.2]
[5]: [4 4.2] [2 3.4] [6 4.8]
[6]: [5 4.8] [3 1.4] [7 1.2]
[7]: [2 5.2] [3 4.3] [6 1.2]

Enter the source vertex and destination vertex to find the shortest path: 0 3
(0, 1) Dist: 2.2
(1, 2) Dist: 1.2
(2, 3) Dist: 6.2

D:\OU\NĂM 2\giaithuat2\graph\x64\Debug\graph.exe (process 20712) exited with code 0.
Press any key to close this window . . .
```