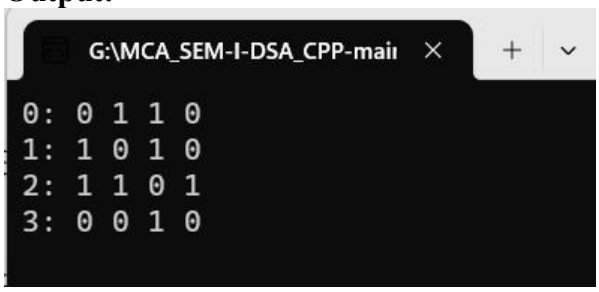
Matrix.cpp:

```
#include<iostream>
using namespace std;
class Graph {
   bool** adjMatrix;
   int numVertices;
public:
   Graph(int numVertices) {
      this->numVertices = numVertices;
      adjMatrix = new bool*[numVertices];
      for (int i = 0; i < numVertices; i++) {
         adjMatrix[i] = new bool[numVertices];
         for (int j = 0; j < numVertices; j++)
            adjMatrix[i][j] = false;
                  }
         }
   void addEdge(int i, int j) {
      adjMatrix[i][j] = true;
      adjMatrix[j][i] = true;
         }
   void removeEdge(int i, int j) {
      adjMatrix[i][j] = false;
      adjMatrix[j][i] = false;
         }
   void toString() {
      for (int i = 0; i < numVertices; i++) {
         cout << i << ": ";
         for (int j = 0; j < numVertices; j++)
            cout << adjMatrix[i][j] << " ";
         cout << "\n";
                  }
         }
   ~Graph() {
      for (int i = 0; i < numVertices; i++)
         delete[] adjMatrix[i];
      delete[] adjMatrix;
         }
};
int main() {
   Graph g(4);
   g.addEdge(0, 1);
   g.addEdge(0, 2);
   g.addEdge(1, 2);
   g.addEdge(2, 0);
   g.addEdge(2, 3);
   g.toString();
   return 0;
}
```

Output:



DirectedGraph.cpp

```
#include<iostream>
#include<conio.h>
#define size 5
using namespace std;
class graph{
public:
    int i, j;
    int wt, u, v, edges;
    int graf[size][size];
    graph();
    void read(int, int, int);
    void display();
    void degree();
};
graph::graph() {
    for (i = 0; i < size; i++) {
        for (j = 0; j < size; j++) {
            graf[i][j] = 0;
        }    }}
void graph::read(int u, int v, int wt) {
    graf[u][v] = wt;
    graf[v][u] = wt;  // Assuming it's an undirected graph
}
void graph::display() {
    cout << "\nGraph is:\n\n";
    for (i = 0; i < size; i++) {
        for (j = 0; j < size; j++) {
            cout << "\t" << graf[i][j];
        }
        cout << endl << endl;
    }}
void graph::degree() {
    int in, out;
    cout << "\nIndegrees & outdegrees of vertices are:\n\n";
    for (i = 0; i < size; i++) {
        in = 0;
        out = 0;
        for (j = 0; j < size; j++) {
            if (graf[i][j] > 0) {
                in++;
            }
            if (graf[j][i] > 0) {
                out++;
            }        }
        cout << "indegree of vertex " << i << " = " << in << endl;
        cout << "outdegree of vertex " << i << " = " << out << endl << endl;
    }}
int main() {
    graph g;
    g.read(2, 3, 1);
    g.display();
    g.degree();
    getch();
    return 0;
}
```

Output:

```
G:\MCA_SEM-I-DSA_CPP-mai

Graph is:

        0       0       0       0       0

        0       0       0       0       0

        0       0       0       1       0

        0       0       1       0       0

        0       0       0       0       0


Indegrees & outdegrees of vertices are:

Indegree of vertex 0 = 0
Outdegree of vertex 0 = 0

Indegree of vertex 1 = 0
Outdegree of vertex 1 = 0

Indegree of vertex 2 = 1
Outdegree of vertex 2 = 1

Indegree of vertex 3 = 1
Outdegree of vertex 3 = 1

Indegree of vertex 4 = 0
Outdegree of vertex 4 = 0
```

Prims.cpp

```
#include<iostream>
#include<conio.h>
#define size 4
using namespace std;
class graph {
public:
    int i;
    int j;
    int wt, u, v, edges;
    int graf[size][size];
    graph();
    void read(int, int, int);
    void display();
    void degree();
    void prim(); // changed the function name from print() to prim()
};
graph::graph() {
    for (i = 0; i < size; i++)
        for (j = 0; j < size; j++)
            graf[i][j] = 0;
}
void graph::read(int u, int v, int wt) {
    graf[u][v] = wt;
    graf[v][u] = wt;
}
void graph::display() {
    cout << "\nGraph is:\n\n";
    for (i = 0; i < size; i++) {
        cout << i << ":";
        for (j = 0; j < size; j++) {
            cout << "\t" << graf[i][j];
        }
        cout << endl << endl;
    }}
void graph::degree() {
    int deg = 0;
    cout << "\nDegrees of vertices are:\n\n";
    for (i = 0; i < size; i++) {
        deg = 0;
        for (j = 0; j < size; j++) {
            if (graf[i][j] > 0)
                deg++;
        }
        cout << "Degree of vertex " << i << " = " << deg << endl;
    }}
void graph::prim() {
    int y, wt = 0, count = 0, min = 0;
    int selected[size];
    for (i = 0; i < size; i++) {
        selected[i] = 0;
    }
    selected[0] = 1;
    cout << "Selected nodes are:\n0";
    while (count < size - 1) {
        min = 9999;
```

```
        for (i = 0; i < size; i++) {
            if (selected[i] == 1) {
                for (j = 0; j < size; j++) {
                    if (selected[j] == 0 && graf[i][j] != 0) {
                        if (min > graf[i][j]) {
                            min = graf[i][j];
                            y = j;
                        } } } } }
        cout << " -> " << y;
        selected[y] = 1;
        wt = wt + graf[u][y];
        count++;
    }
    cout << "\nWeight is: " << wt;
}

int main() {
    graph g;
    g.read(0, 1, 2);
    g.read(0, 1, 7);
    g.read(1, 9, 6);
    g.read(1, 2, 5);
    g.display();
    g.degree();
    g.prim(); // changed the function name from print() to prim()
    getch();
    return 0;
}
```

Output:

```
G:\MCA_SEM-I-DSA_CPP-mai

Graph is:

0:      0       7       0       0

1:      7       0       5       0

2:      0       5       0       0

3:      0       6       0       0


Degrees of vertices are:

Degree of vertex 0 = 1
Degree of vertex 1 = 2
Degree of vertex 2 = 1
Degree of vertex 3 = 1
Selected nodes are:
0 -> 1 -> 2 -> 2
Weight is: 7
```