

# KATHMANDU UNIVERSITY



## REPORT : LAB 6

### **Submitted By:**

Name: Bikraj Shrestha  
Roll:50

Name: Janak Sitaula  
Roll:51

*Group : CE  
2nd Year/1st Sem*

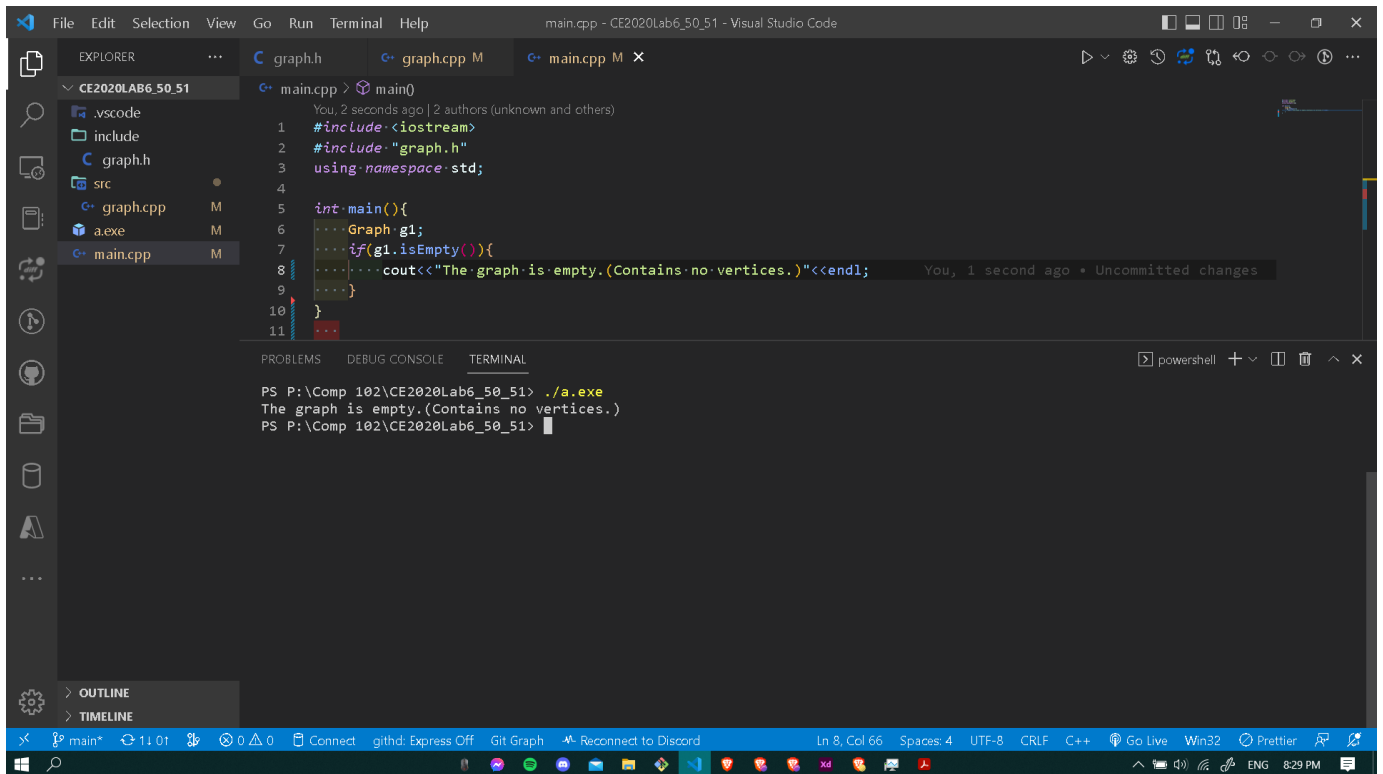
### **Submitted To:**

*Dr. Rajani Chulyadyo  
Undergraduate Coordinator CE  
Program  
Department of Computer Science  
and Engineering*

# GRAPH

## 1.isEmpty()

*This function checks if the graph is empty. It checks the size of the vertices by counting all the vertex in the vertices.*



```
File Edit Selection View Go Run Terminal Help
main.cpp - CE2020Lab6_50_51 - Visual Studio Code

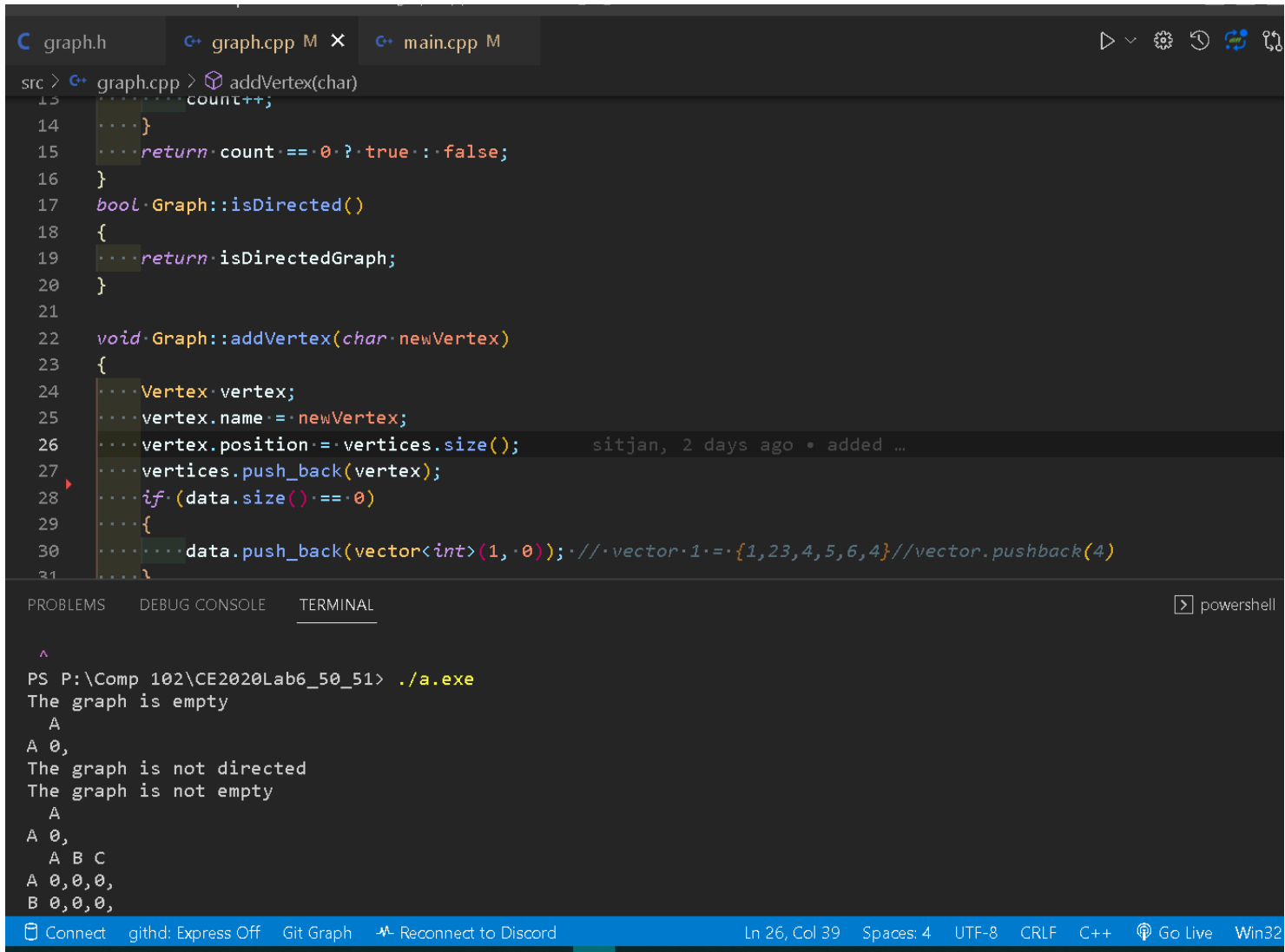
EXPLORER
CE2020Lab6_50_51
  .vscode
  include
  graph.h
  src
    graph.cpp M
    a.exe M
    main.cpp M

main.cpp > main()
You, 2 seconds ago | 2 authors (unknown and others)
1 #include <iostream>
2 #include "graph.h"
3 using namespace std;
4
5 int main(){
6     Graph g1;
7     if(g1.isEmpty()){
8         cout<<"The graph is empty.(Contains no vertices.)"<<endl;
9     }
10 }
11

PROBLEMS DEBUG CONSOLE TERMINAL
PS P:\Comp 102\CE2020Lab6_50_51> ./a.exe
The graph is empty.(Contains no vertices.)
PS P:\Comp 102\CE2020Lab6_50_51>
```

## 2.isDirected()

*This function checks if the graph is directed. The member variable called isDirected is returned that is responsible for tracking if the graph is directed or not.*



```
graph.h  graph.cpp M  main.cpp M
src > graph.cpp > addVertex(char)
13     count++;
14     ...}
15     return count == 0 ? true : false;
16 }
17 bool Graph::isDirected()
18 {
19     return isDirectedGraph;
20 }
21
22 void Graph::addVertex(char newVertex)
23 {
24     Vertex vertex;
25     vertex.name = newVertex;
26     vertex.position = vertices.size();
27     vertices.push_back(vertex);
28     if (data.size() == 0)
29     {
30         data.push_back(vector<int>(1, 0)); // vector 1 = {1,2,3,4,5,6,4} // vector.pushback(4)
31     }
32 }
```

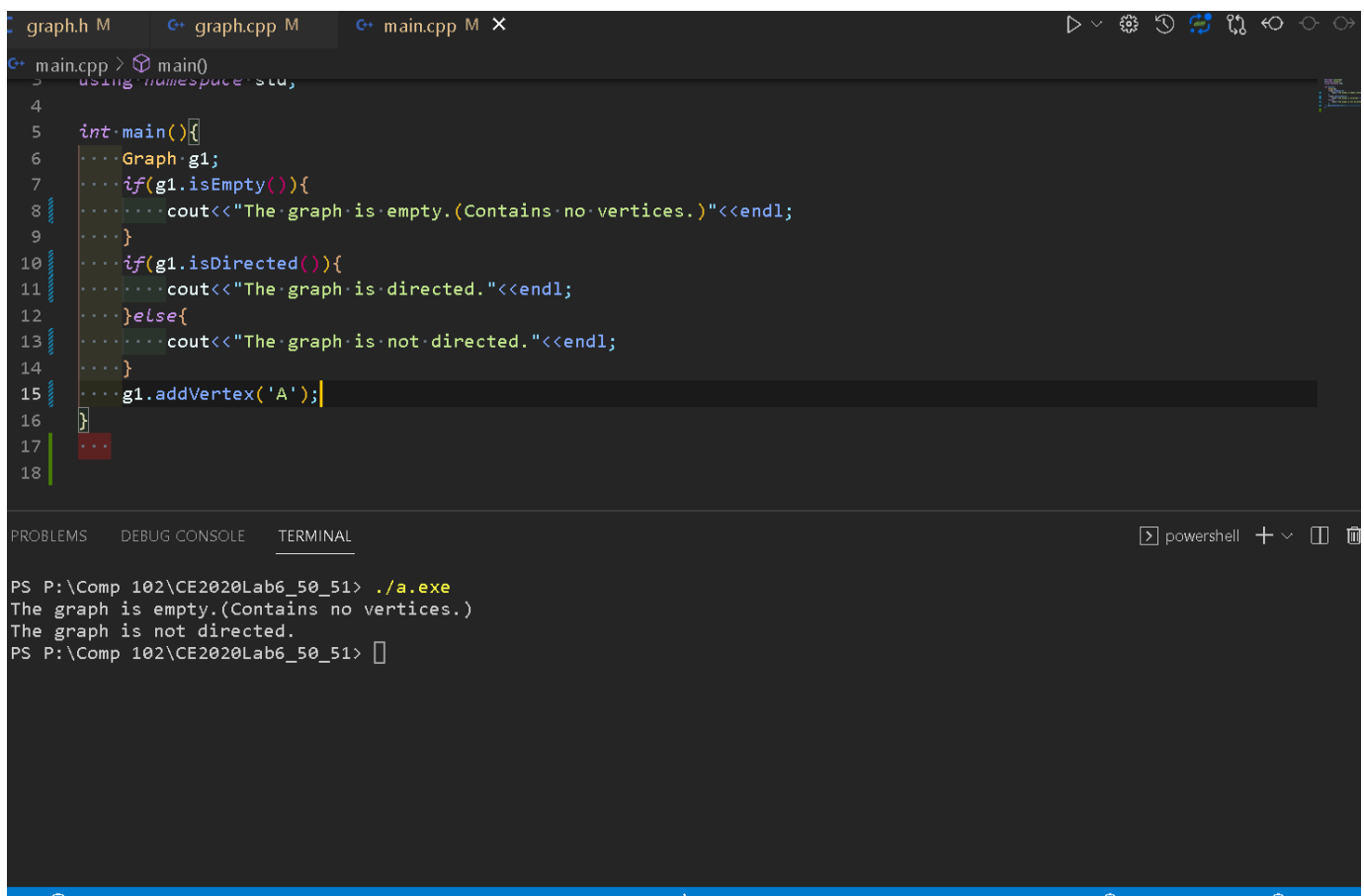
PROBLEMS DEBUG CONSOLE TERMINAL powershell

```
PS P:\Comp 102\CE2020Lab6_50_51> ./a.exe
The graph is empty
A
A 0,
The graph is not directed
The graph is not empty
A
A 0,
A B C
A 0,0,0,
B 0,0,0,
```

Ln 26, Col 39 Spaces: 4 UTF-8 CRLF C++ Go Live Win32

### 3.addVertex(char vertex1)

*The function adds a vertex to the graph. At first the vertex is added to the vertices vector that is responsible for tracking the vertices. Then the size of the data vector is increased by one (initialized with zeros.). The vertex added to the vertices has a name member that stores the name of the vertex. For eg: 'A'.*



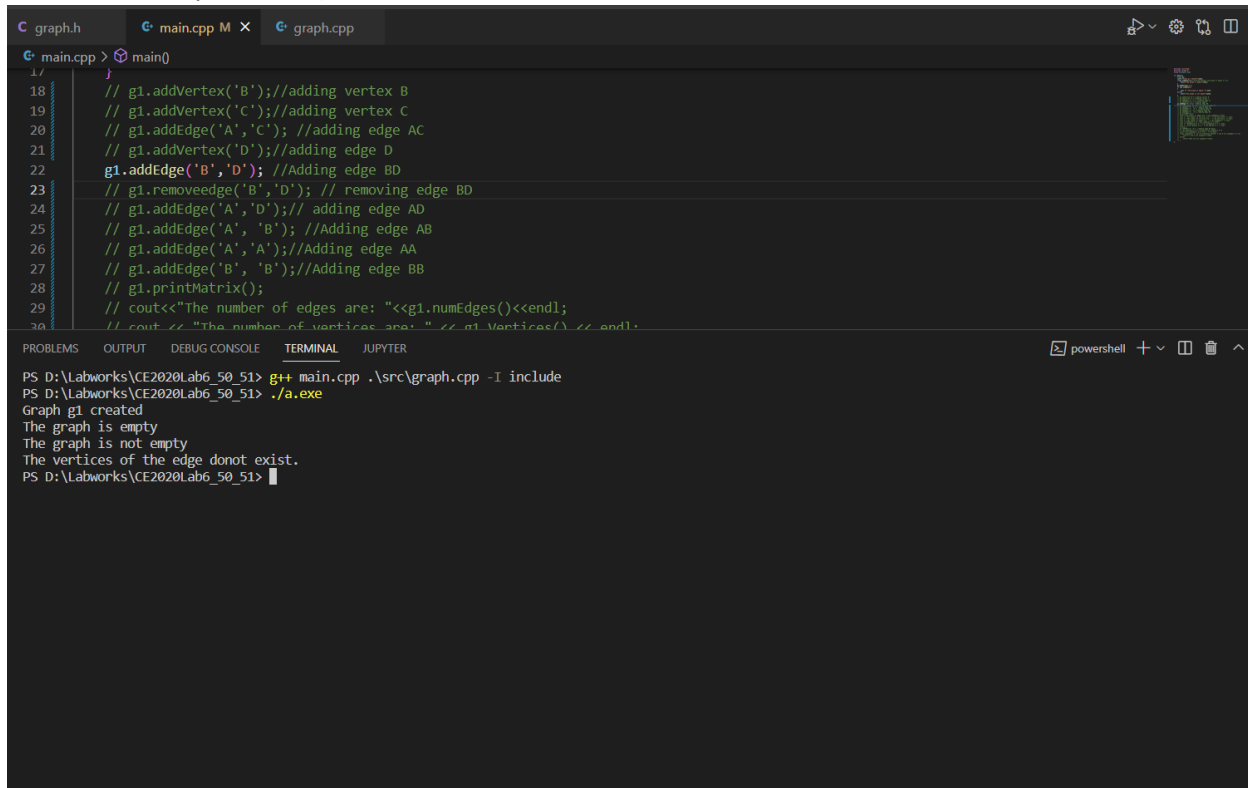
```
graph.h M  graph.cpp M  main.cpp M X
main.cpp > main()
using namespace std;
4
5 int main(){
6     Graph g1;
7     if(g1.isEmpty()){
8         cout<<"The graph is empty.(Contains no vertices.)"<<endl;
9     }
10    if(g1.isDirected()){
11        cout<<"The graph is directed."<<endl;
12    }else{
13        cout<<"The graph is not directed."<<endl;
14    }
15    g1.addVertex('A');
16 }
17
18
```

PROBLEMS DEBUG CONSOLE TERMINAL powershell + -

```
PS P:\Comp 102\CE2020Lab6_50_51> ./a.exe
The graph is empty.(Contains no vertices.)
The graph is not directed.
PS P:\Comp 102\CE2020Lab6_50_51>
```

#### 4. addEdge(vertex1, vertex2): Adds an edge from vertex1 to vertex2

This function adds edge in the graph. The added graph will have value on its adjacency matrix (can be seen on the output matrix).



The screenshot shows a C++ IDE with three files: graph.h, main.cpp, and graph.cpp. The main.cpp file contains the following code:

```
17 // main()
18 // g1.addVertex('B');//adding vertex B
19 // g1.addVertex('C');//adding vertex C
20 // g1.addEdge('A','C'); //adding edge AC
21 // g1.addVertex('D');//adding edge D
22 g1.addEdge('B','D'); //Adding edge BD
23 // g1.removeEdge('B','D'); // removing edge BD
24 // g1.addEdge('A','D');// adding edge AD
25 // g1.addEdge('A','B'); //Adding edge AB
26 // g1.addEdge('A','A');//Adding edge AA
27 // g1.addEdge('B','B');//Adding edge BB
28 // g1.printMatrix();
29 // cout<<"The number of edges are: "<<g1.numEdges()<<endl;
30 // cout<<"The number of vertices are: "<<g1.Vertices()<<endl;
```

The terminal output shows the following commands and results:

```
PS D:\Labworks\CE2020Lab6_50_51> g++ main.cpp .\src\graph.cpp -I include
PS D:\Labworks\CE2020Lab6_50_51> ./a.exe
Graph g1 created
The graph is empty
The graph is not empty
The vertices of the edge donot exist.
PS D:\Labworks\CE2020Lab6_50_51>
```

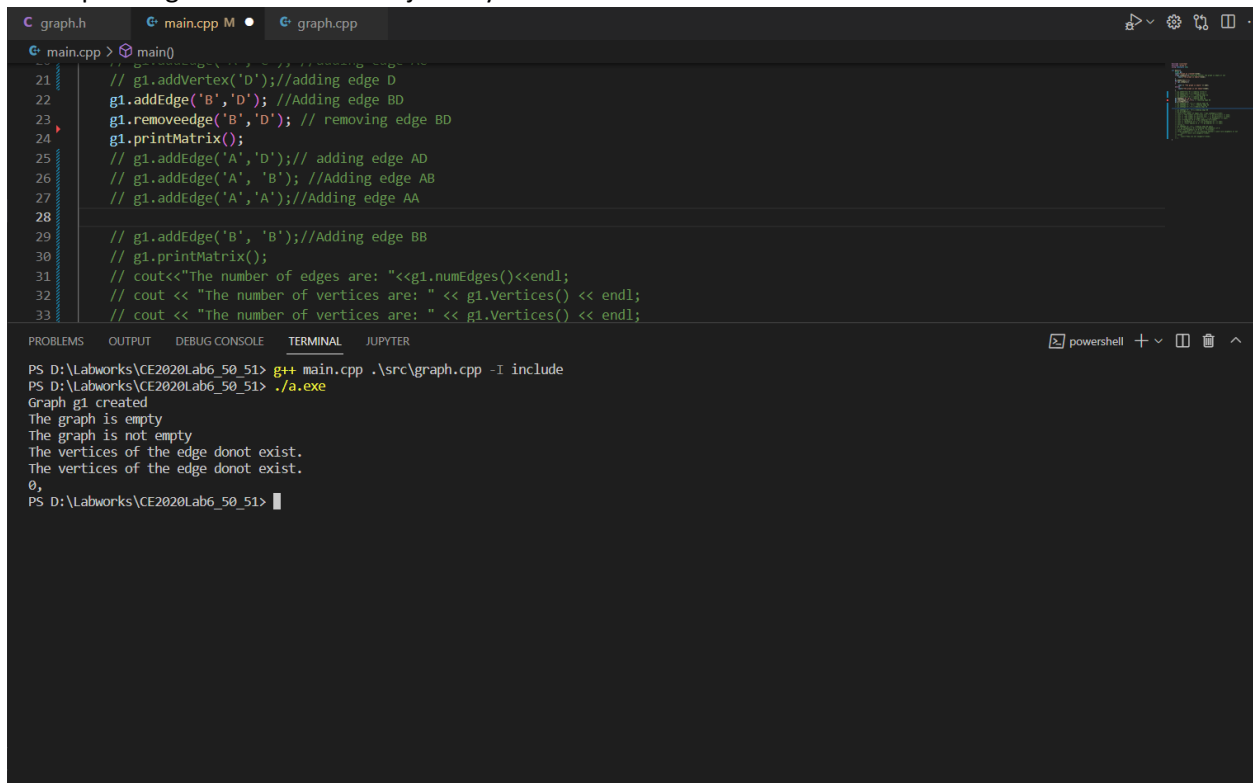
Figure 1 Here the output says that the vertices of the edge do not exist because we have not added any vertex and directly tried to add an edge which is not possible

#### 5. removeVertex(vertexToRemove): Remove a vertex from the graph

This function removes the desired vertex from the graph. This can be seen when all the corresponding edges are removed and denoted by zero in the matrix. The value for any assigned vertex will be assigned as 'z'.

## 6. removeEdge(vertex1, vertex2): Remove an edge from the graph

This function removes edge from the graph. This is done by assigning zeroes to the desired edges corresponding elements in the adjacency matrix.

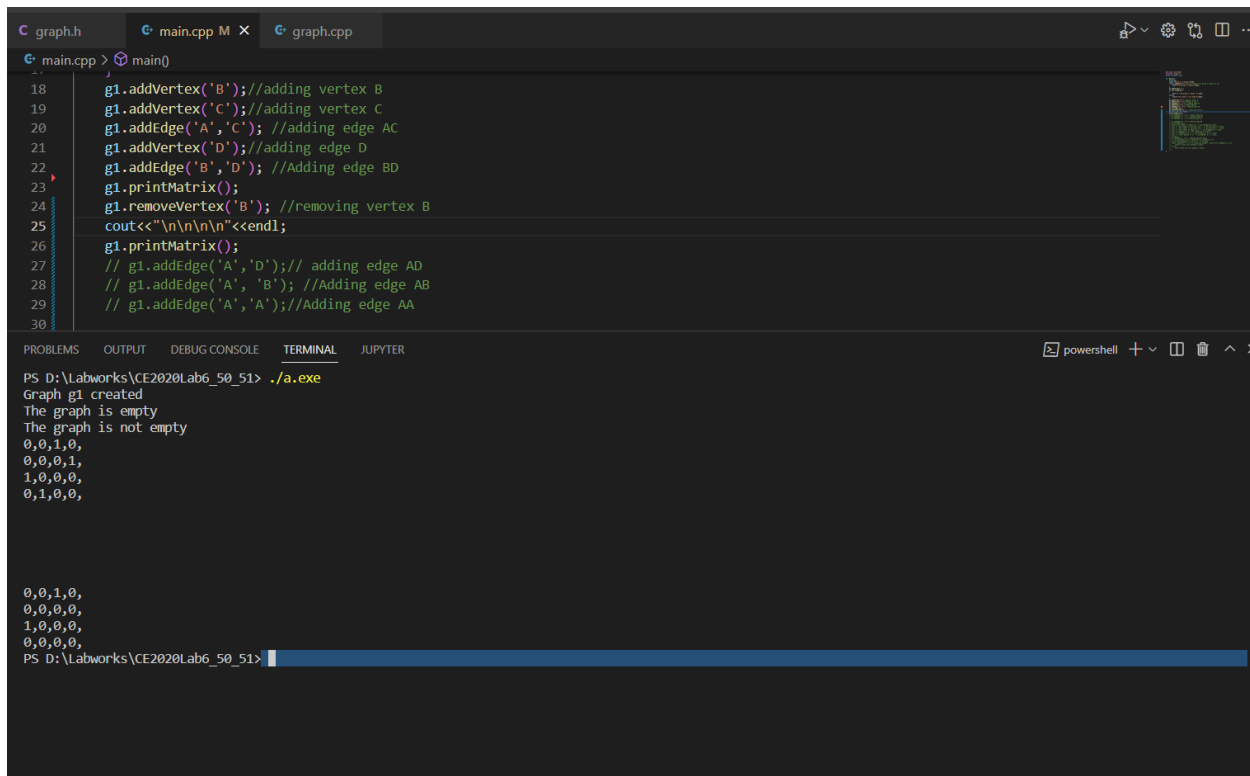


```
graph.h  main.cpp M  graph.cpp
main.cpp > main()
21 // g1.addVertex('D');//adding edge D
22 g1.addEdge('B','D');//Adding edge BD
23 g1.removeEdge('B','D');// removing edge BD
24 g1.printMatrix();
25 // g1.addEdge('A','D');// adding edge AD
26 // g1.addEdge('A', 'B');//Adding edge AB
27 // g1.addEdge('A','A');//Adding edge AA
28
29 // g1.addEdge('B', 'B');//Adding edge BB
30 // g1.printMatrix();
31 // cout<<"The number of edges are: "<<g1.numEdges()<<endl;
32 // cout << "The number of vertices are: " << g1.Vertices() << endl;
33 // cout << "The number of vertices are: " << g1.Vertices() << endl;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
PS D:\Labworks\CE2020Lab6_50_51> g++ main.cpp .\src\graph.cpp -I include
PS D:\Labworks\CE2020Lab6_50_51> ./a.exe
Graph g1 created
The graph is empty
The graph is not empty
The vertices of the edge donot exist.
The vertices of the edge donot exist.
0,
PS D:\Labworks\CE2020Lab6_50_51>
```

Figure 2 Here the output says the vertices of the edge do not exist as we have not added any vertices but tried to add and remove an edge



```
graph.h  main.cpp M  graph.cpp
main.cpp > main()
18     g1.addVertex('B');//adding vertex B
19     g1.addVertex('C');//adding vertex C
20     g1.addEdge('A','C'); //adding edge AC
21     g1.addVertex('D');//adding edge D
22     g1.addEdge('B','D'); //Adding edge BD
23     g1.printMatrix();
24     g1.removeVertex('B'); //removing vertex B
25     cout<<"\n\n\n\n"<<endl;
26     g1.printMatrix();
27     // g1.addEdge('A','D');// adding edge AD
28     // g1.addEdge('A','B'); //Adding edge AB
29     // g1.addEdge('A','A');//Adding edge AA
30 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
PS D:\Labworks\CE2020Lab6_50_51> ./a.exe
Graph g1 created
The graph is empty
The graph is not empty
0,0,1,0,
0,0,0,1,
1,0,0,0,
0,1,0,0,

0,0,1,0,
0,0,0,0,
1,0,0,0,
0,0,0,0,
PS D:\Labworks\CE2020Lab6_50_51>
```

Figure 3 Here we can see as the vertex and edge have been added and removed, the corresponding matrix show their existence with 1's and 0's

## 7. numVertices(): Returns the number of vertices in the graph

This function returns the total number of vertices in the graph. This is done by counting the total number of alphabets in the data. This excludes 'z', which is assigned for any removed vertex in the graph.

## 8. numEdges(): Returns the number of edges in the graph

This function returns the total number of edges in the graph. This is performed by calculating the total number of 1's in the adjacency matrix in the data.

```
24 g1.removeVertex('D'); //removing vertex D
25 cout<<"\n\n\n\n"<<endl;
26 g1.printMatrix();
27 g1.removeEdge('A','C');
28 g1.addEdge('A','D');// adding edge AD
29 g1.addEdge('A','B');//Adding edge AB
30 g1.addEdge('A','A');//Adding edge AA
31 g1.addEdge('B','B');//Adding edge BB
32 g1.printMatrix();
33 cout<<"The number of edges are: "<<g1.numEdges()<<endl;
34 cout << "The number of vertices are: " << g1.Vertices() << endl;
35 cout << "The number of vertices are: " << g1.Vertices() << endl;
36 // cout << "The number of edges are: " << g1.numEdges() << endl;
37 // cout << "InDegree of A " << g1.indegree('A')<<endl;
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** JUPYTER

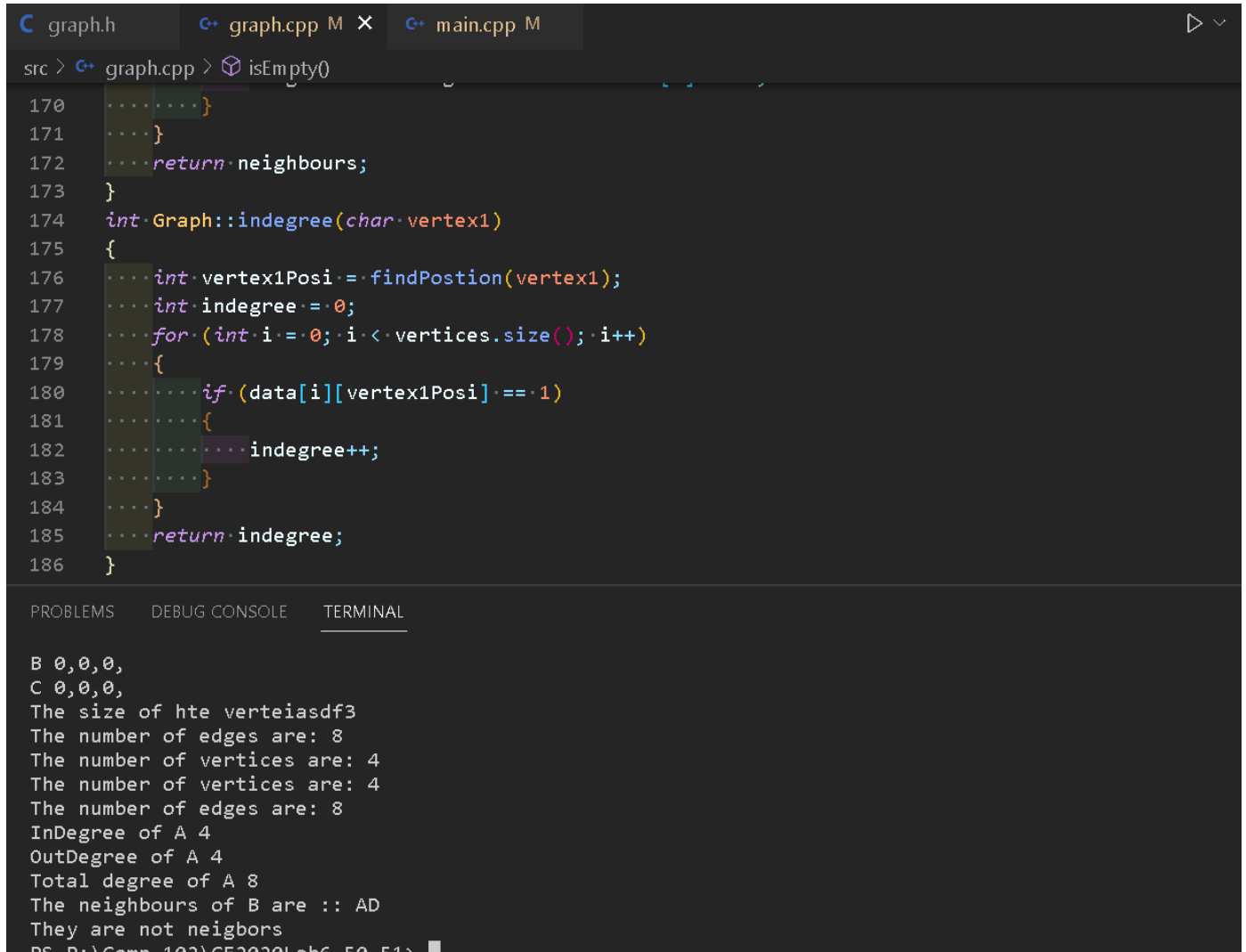
Graph g1 created  
The graph is empty  
The graph is not empty  
0,0,1,0,  
0,0,0,1,  
1,0,0,0,  
0,1,0,0,  
  
0,0,1,0,  
0,0,0,0,  
1,0,0,0,  
0,0,0,0,  
The vertices of the edge donot exist.  
The vertices of the edge donot exist.  
1,0,0,1,  
0,0,0,0,  
0,0,0,0,  
1,0,0,0,  
The number of edges are: 3  
The number of vertices are: 3  
The number of vertices are: 3  
PS D:\Labworks\CE2020Lab6 50 51>

Figure 4 Here we can see all the functions in work together with their respective adjacency matrices.



## 9.inDegree()

*The function returns the number of the edge that are coming out of the vertex. A loop runs to check all the 1's present in the row of the adjacency matrix. And that value is returned.*



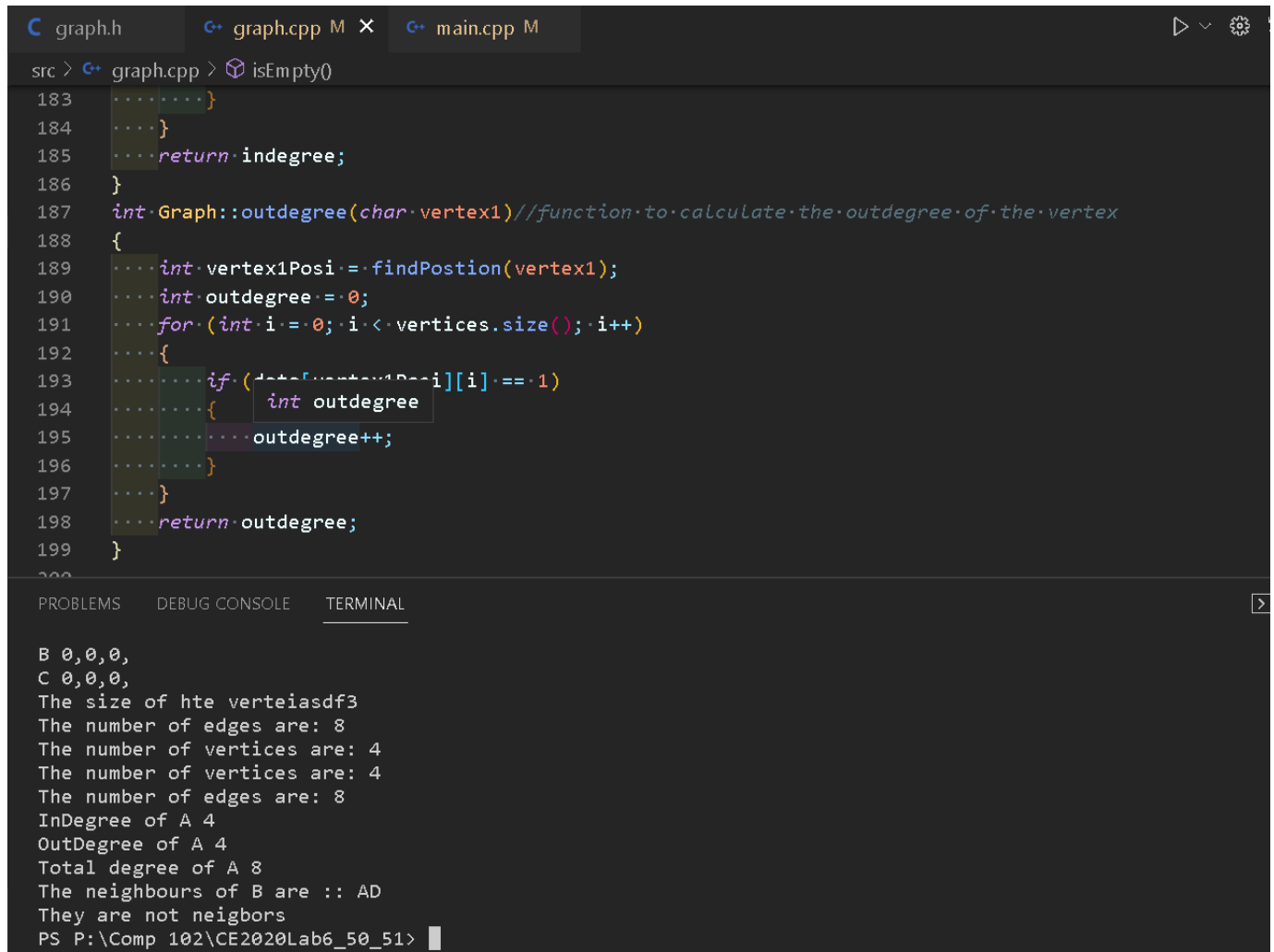
```
graph.h  graph.cpp M  main.cpp M
src > graph.cpp > isEmpty()
170     ....}
171     ....}
172     ....return neighbours;
173 }
174 int Graph::indegree(char vertex1)
175 {
176     ....int vertex1Posi=findPostion(vertex1);
177     ....int indegree=0;
178     ....for(int i=0;i<vertices.size();i++)
179     ....{
180     ....    ....if(data[i][vertex1Posi]==1)
181     ....    ....{
182     ....    ....    ....indegree++;
183     ....    ....}
184     ....}
185     ....return indegree;
186 }
```

PROBLEMS DEBUG CONSOLE TERMINAL

```
B 0,0,0,
C 0,0,0,
The size of hte verteiasdf3
The number of edges are: 8
The number of vertices are: 4
The number of vertices are: 4
The number of edges are: 8
InDegree of A 4
OutDegree of A 4
Total degree of A 8
The neighbours of B are :: AD
They are not neighbors
PS: D:\Comp_102\CE3020Lab6_50_51>
```

## 9.outDegree()

*The function returns the number of the edge that are coming out of the vertex. A loop runs to check all the 1's present in the column of the adjacency matrix. And that value is returned.*

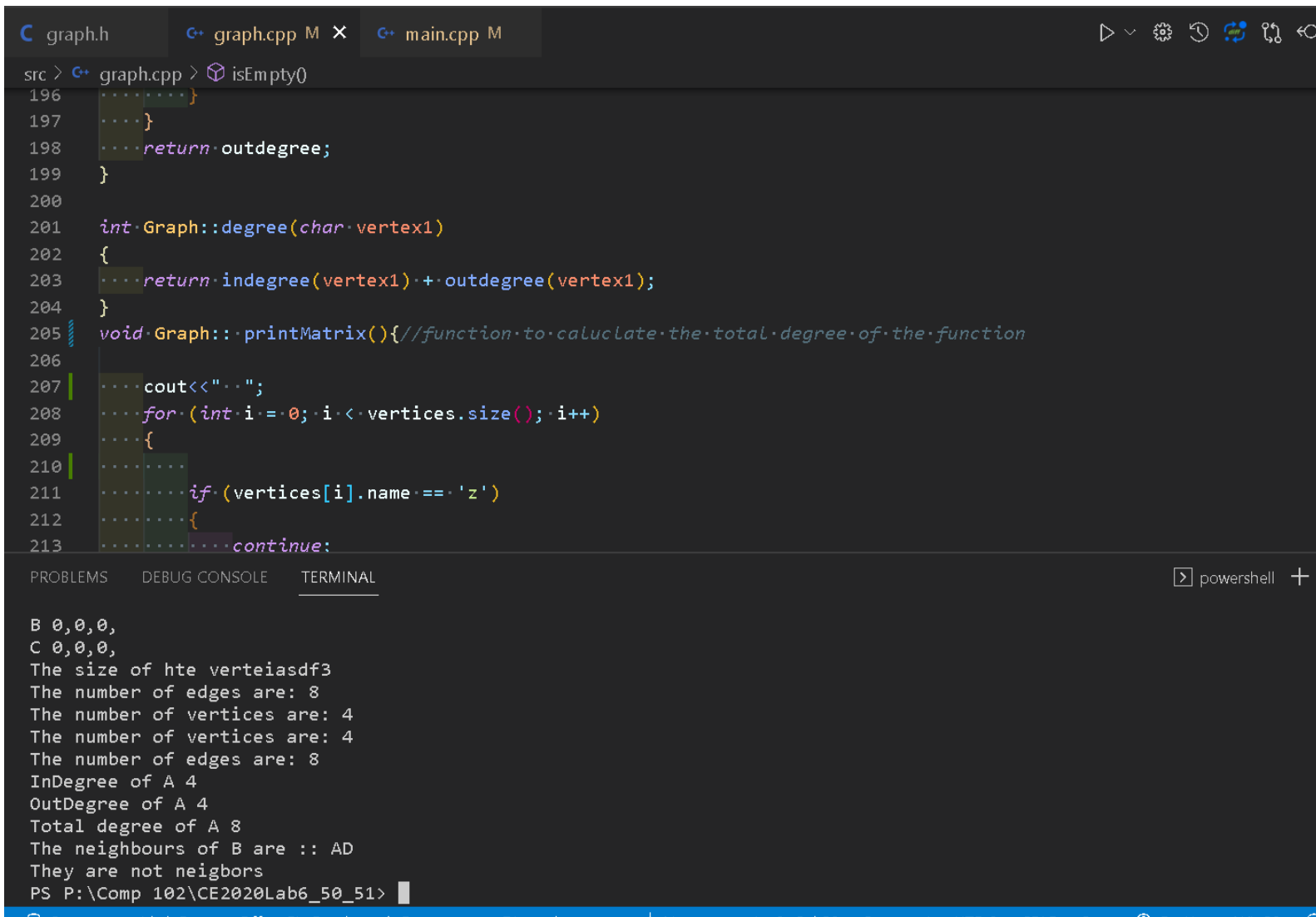


```
graph.h  graph.cpp M  main.cpp M
src > graph.cpp > isEmpty()
183     ....}
184     ....}
185     ....return indegree;
186 }
187 int Graph::outdegree(char vertex1)//function to calculate the outdegree of the vertex
188 {
189     ....int vertex1Posi=findPostion(vertex1);
190     ....int outdegree=0;
191     ....for(int i=0;i<vertices.size();i++)
192     ....{
193     ....    ....if(data[vertex1Posi][i]==1)
194     ....    ....{ int outdegree
195     ....    ....    ....outdegree++;
196     ....    ....}
197     ....}
198     ....return outdegree;
199 }
200

PROBLEMS  DEBUG CONSOLE  TERMINAL
B 0,0,0,
C 0,0,0,
The size of hte verteiasdf3
The number of edges are: 8
The number of vertices are: 4
The number of vertices are: 4
The number of edges are: 8
InDegree of A 4
OutDegree of A 4
Total degree of A 8
The neighbours of B are :: AD
They are not neighbors
PS P:\Comp 102\CE2020Lab6_50_51>
```

## 11.degree()char vertex)

The function returns the sum of indegree and outdegree of the vertex.



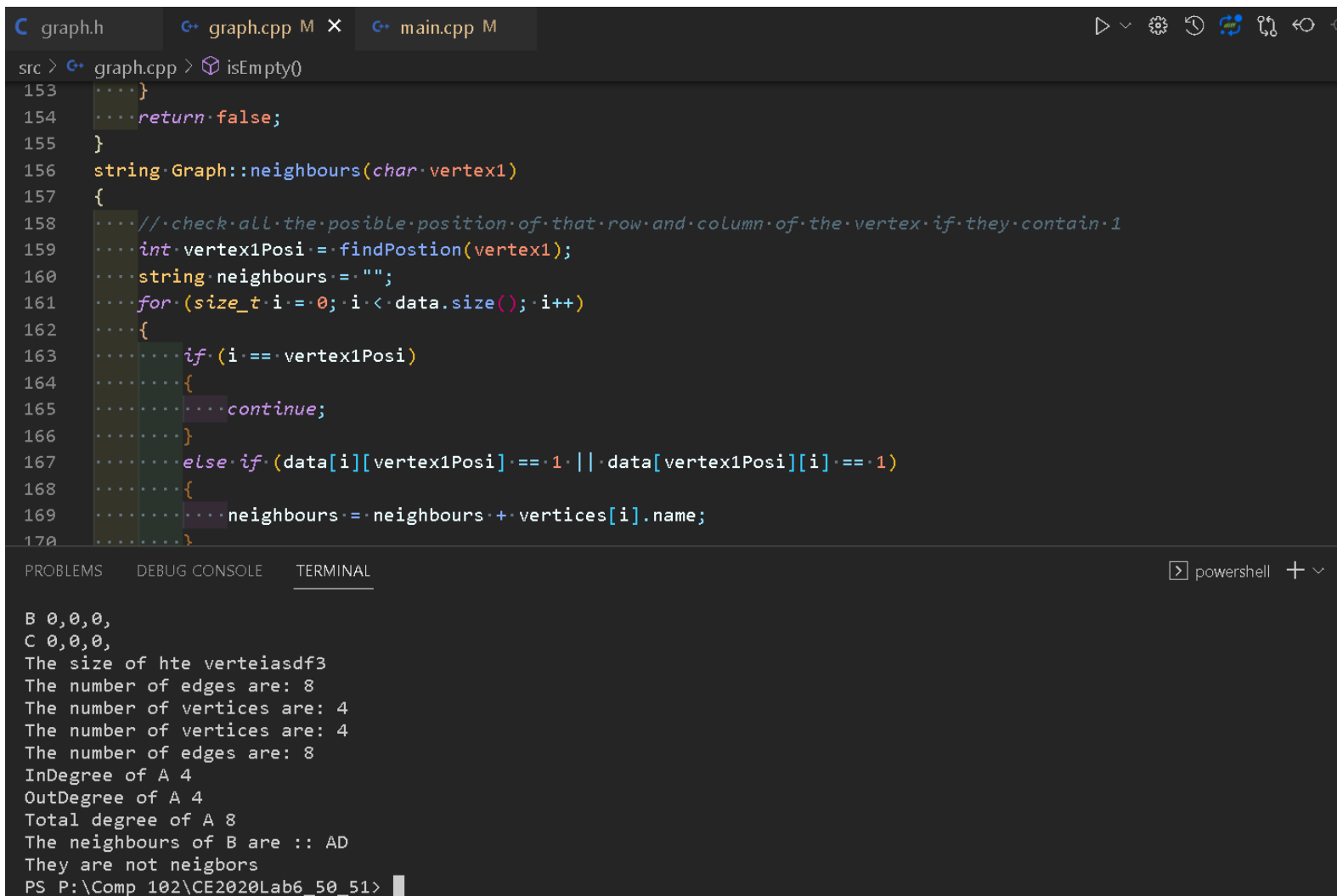
```
graph.h  graph.cpp M x  main.cpp M
src > graph.cpp > isEmpty()
196     }
197 }
198 return outdegree;
199 }
200
201 int Graph::degree(char vertex1)
202 {
203     return indegree(vertex1) + outdegree(vertex1);
204 }
205 void Graph::printMatrix(){//function to calculate the total degree of the function
206
207     cout<<" ";
208     for(int i=0; i<vertices.size(); i++)
209     {
210
211         if(vertices[i].name=='z')
212         {
213             continue;
```

PROBLEMS DEBUG CONSOLE TERMINAL powershell +

```
B 0,0,0,
C 0,0,0,
The size of hte verteiasdf3
The number of edges are: 8
The number of vertices are: 4
The number of vertices are: 4
The number of edges are: 8
InDegree of A 4
OutDegree of A 4
Total degree of A 8
The neighbours of B are :: AD
They are not neighbors
PS P:\Comp 102\CE2020Lab6_50_51>
```

## 12.neighbors(char vertex1)

*The function returns all the neighbors of the vertex. The row and the column of the given vertex is checked if any of them have 1. If 1 is found then , it is a neighbors and the value is returned.*



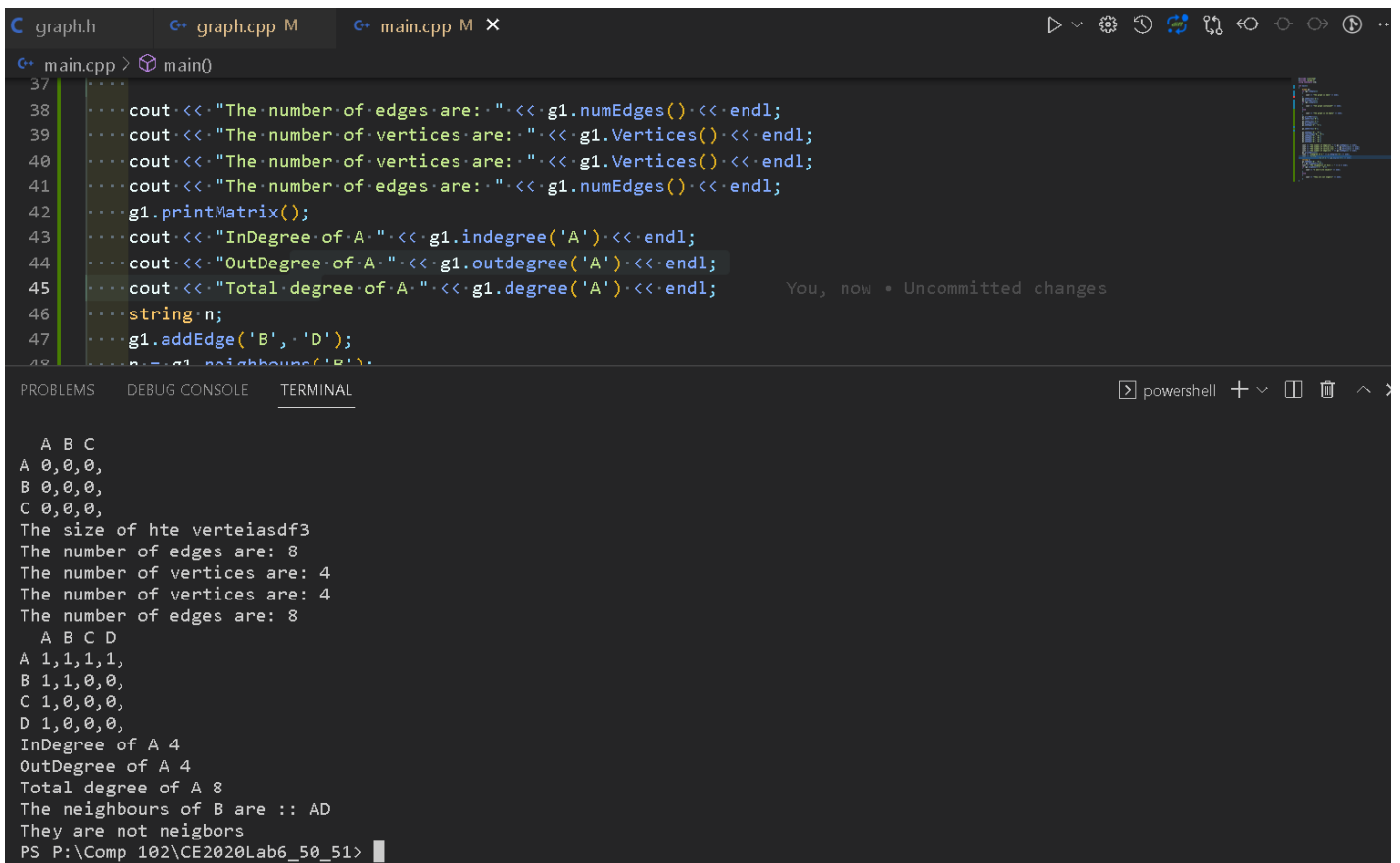
```
graph.h  graph.cpp M  main.cpp M
src > graph.cpp > isEmpty()
153     }
154     return false;
155 }
156 string Graph::neighbours(char vertex1)
157 {
158     //check all the possible position of that row and column of the vertex if they contain 1
159     int vertex1Posi = findPosition(vertex1);
160     string neighbours = "";
161     for (size_t i = 0; i < data.size(); i++)
162     {
163         if (i == vertex1Posi)
164         {
165             continue;
166         }
167         else if (data[i][vertex1Posi] == 1 || data[vertex1Posi][i] == 1)
168         {
169             neighbours = neighbours + vertices[i].name;
170         }
171     }
172 }
```

PROBLEMS DEBUG CONSOLE TERMINAL

```
B 0,0,0,
C 0,0,0,
The size of hte verteiasdf3
The number of edges are: 8
The number of vertices are: 4
The number of vertices are: 4
The number of edges are: 8
InDegree of A 4
OutDegree of A 4
Total degree of A 8
The neighbours of B are :: AD
They are not neighbors
PS P:\Comp 102\CE2020Lab6_50_51>
```

### 13.neighbors(char vertex1 , char vertex2)

*This function checks if the an edge runs through the vertices.  
For this the possible edge i.e veretx1Vertex2 or vertex2Vertex1  
is checked. If it exists then they are neighbors.*



```
graph.h  graph.cpp M  main.cpp M X
main.cpp > main()
37  ....
38  ....cout<<<"The number of edges are:"<<<g1.numEdges()<<<endl;
39  ....cout<<<"The number of vertices are:"<<<g1.Vertices()<<<endl;
40  ....cout<<<"The number of vertices are:"<<<g1.Vertices()<<<endl;
41  ....cout<<<"The number of edges are:"<<<g1.numEdges()<<<endl;
42  ....g1.printMatrix();
43  ....cout<<<"InDegree of A:"<<<g1.indegree('A')<<<endl;
44  ....cout<<<"OutDegree of A:"<<<g1.outdegree('A')<<<endl;
45  ....cout<<<"Total degree of A:"<<<g1.degree('A')<<<endl;
46  ....string n;
47  ....g1.addEdge('B','D');
48  ....n=g1.neighbors('B');

PROBLEMS  DEBUG CONSOLE  TERMINAL
powershell + - [ ] [ ] ^ >

A B C
A 0,0,0,
B 0,0,0,
C 0,0,0,
The size of hte verteiasdf3
The number of edges are: 8
The number of vertices are: 4
The number of vertices are: 4
The number of edges are: 8
A B C D
A 1,1,1,1,
B 1,1,0,0,
C 1,0,0,0,
D 1,0,0,0,
InDegree of A 4
OutDegree of A 4
Total degree of A 8
The neighbours of B are :: AD
They are not neighbors
PS P:\Comp 102\CE2020Lab6_50_51>
```