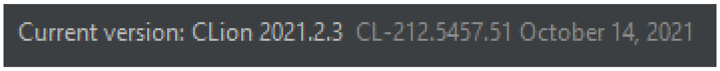
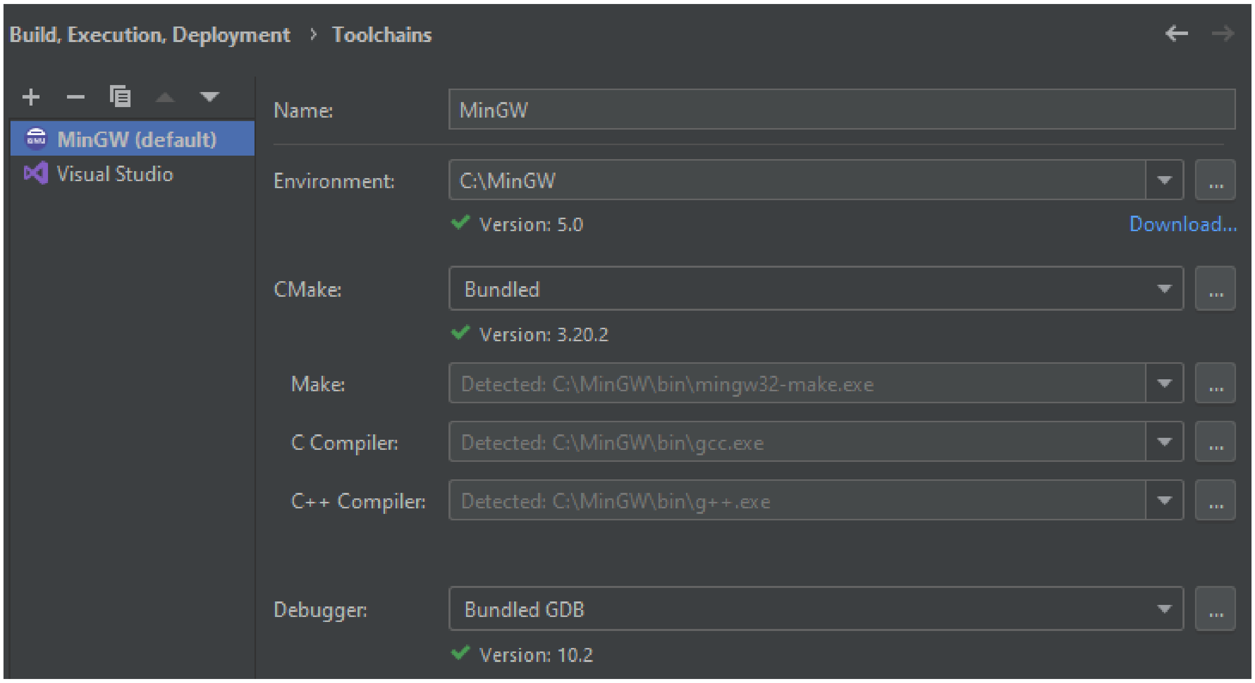
This program was made and tested in CLion version 2021.2.3:



Current version: CLion 2021.2.3 CL-212.5457.51 October 14, 2021

Using the MinGW toolchain:



Name: MinGW

Environment: C:\MinGW

✓ Version: 5.0

CMake: Bundled

✓ Version: 3.20.2

Make: Detected: C:\MinGW\bin\mingw32-make.exe

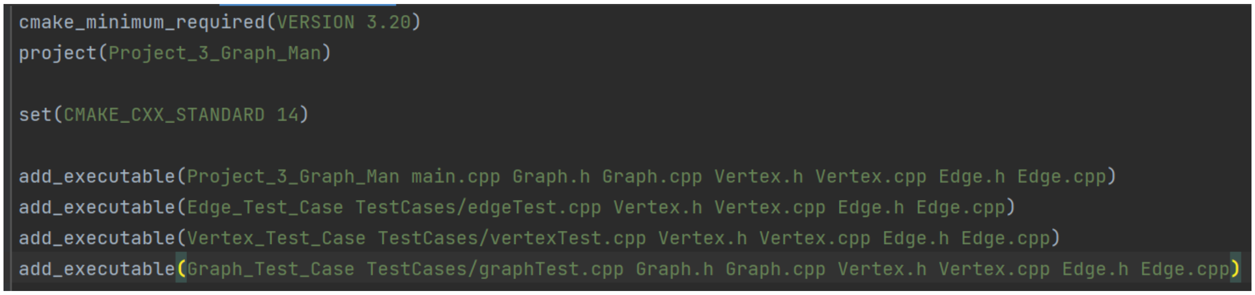
C Compiler: Detected: C:\MinGW\bin\gcc.exe

C++ Compiler: Detected: C:\MinGW\bin\g++.exe

Debugger: Bundled GDB

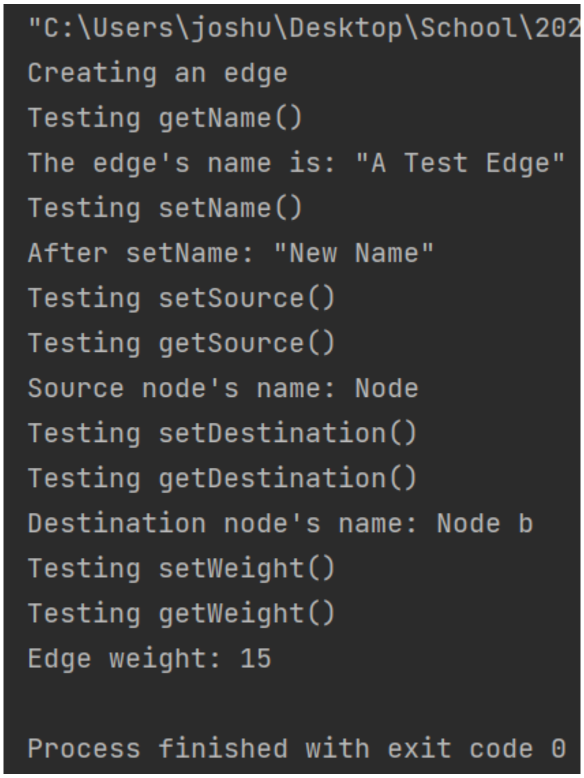
✓ Version: 10.2

My CMakeLists.txt:



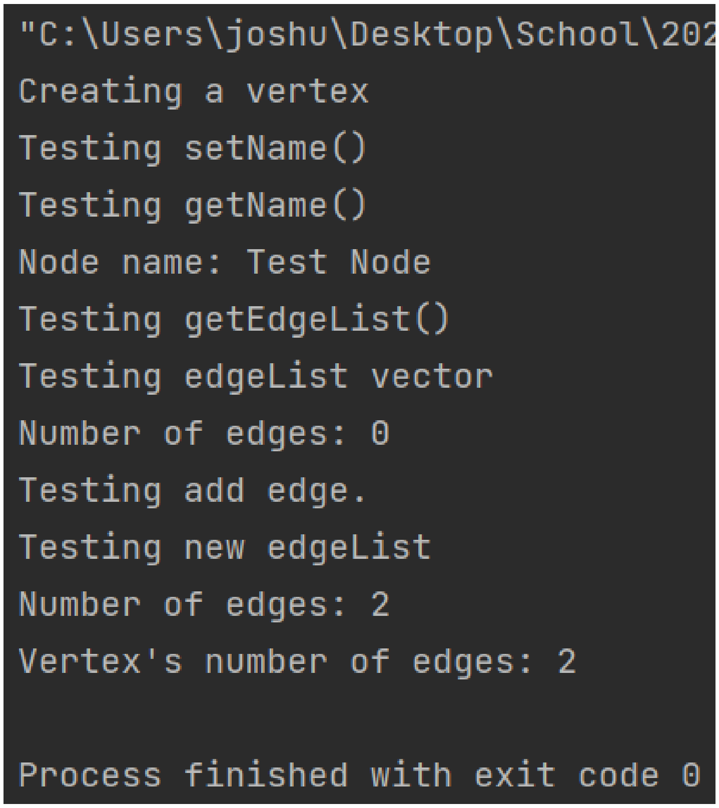
```
cmake_minimum_required(VERSION 3.20)
project(Project_3_Graph_Man)

set(CMAKE_CXX_STANDARD 14)

add_executable(Project_3_Graph_Man main.cpp Graph.h Graph.cpp Vertex.h Vertex.cpp Edge.h Edge.cpp)
add_executable(Edge_Test_Case TestCases/edgeTest.cpp Vertex.h Vertex.cpp Edge.h Edge.cpp)
add_executable(Vertex_Test_Case TestCases/vertexTest.cpp Vertex.h Vertex.cpp Edge.h Edge.cpp)
add_executable(Graph_Test_Case TestCases/graphTest.cpp Graph.h Graph.cpp Vertex.h Vertex.cpp Edge.h Edge.cpp)
```

edgeTest.cpp output:



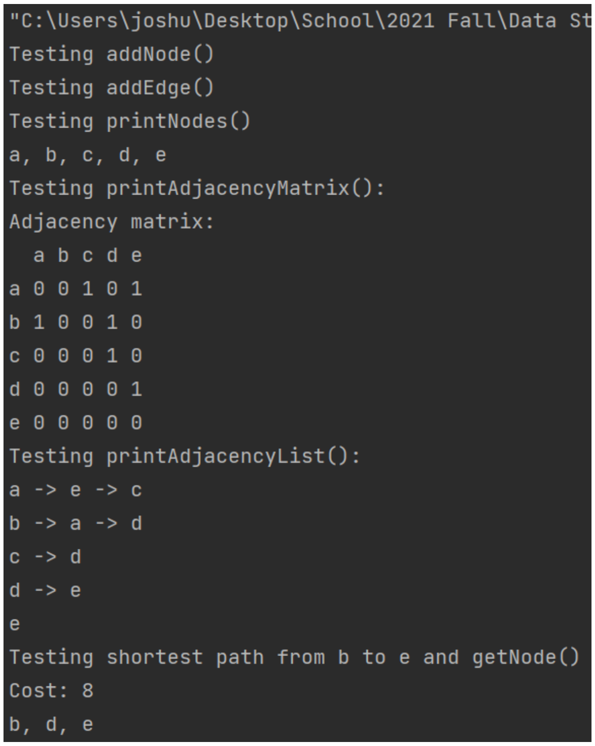
```
"C:\Users\joshu\Desktop\School\202
Creating an edge
Testing getName()
The edge's name is: "A Test Edge"
Testing setName()
After setName: "New Name"
Testing setSource()
Testing getSource()
Source node's name: Node
Testing setDestination()
Testing getDestination()
Destination node's name: Node b
Testing setWeight()
Testing getWeight()
Edge weight: 15

Process finished with exit code 0
```

vertextTest.cpp output:

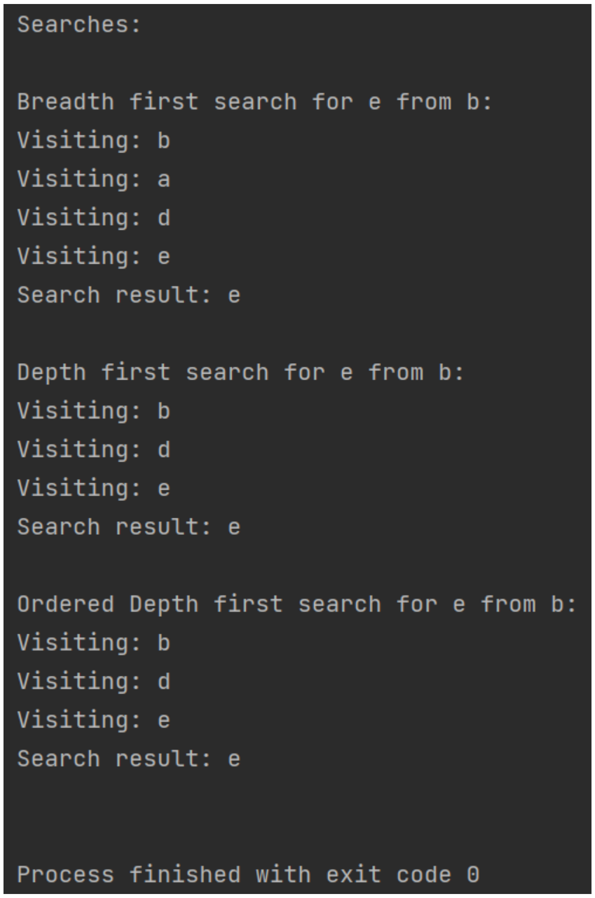


```
"C:\Users\joshu\Desktop\School\202
Creating a vertex
Testing setName()
Testing getName()
Node name: Test Node
Testing getEdgeList()
Testing edgeList vector
Number of edges: 0
Testing add edge.
Testing new edgeList
Number of edges: 2
Vertex's number of edges: 2

Process finished with exit code 0
```

graphTestCase.cpp output:



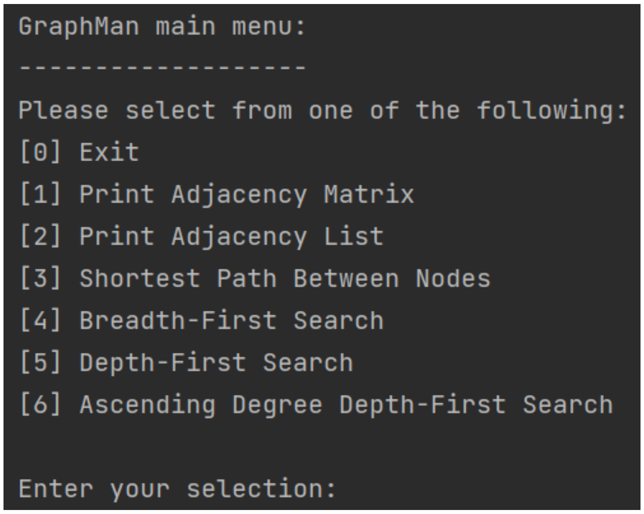
```
"C:\Users\joshu\Desktop\School\2021 Fall\Data St
Testing addNode()
Testing addEdge()
Testing printNodes()
a, b, c, d, e
Testing printAdjacencyMatrix():
Adjacency matrix:
  a b c d e
a 0 0 1 0 1
b 1 0 0 1 0
c 0 0 0 1 0
d 0 0 0 0 1
e 0 0 0 0 0
Testing printAdjacencyList():
a -> e -> c
b -> a -> d
c -> d
d -> e
e
Testing shortest path from b to e and getNode()
Cost: 8
b, d, e
```



```
Searches:

Breadth first search for e from b:
Visiting: b
Visiting: a
Visiting: d
Visiting: e
Search result: e

Depth first search for e from b:
Visiting: b
Visiting: d
Visiting: e
Search result: e

Ordered Depth first search for e from b:
Visiting: b
Visiting: d
Visiting: e
Search result: e



Process finished with exit code 0
```

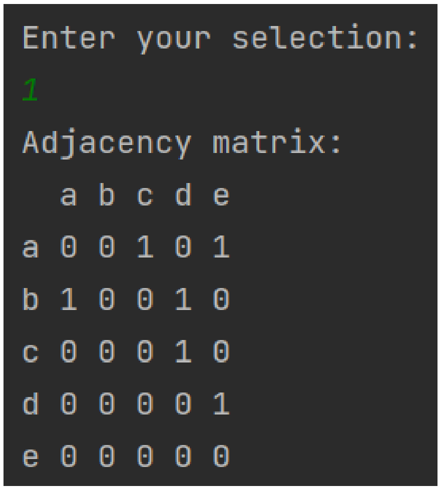
main.cpp output:

The main menu of the program



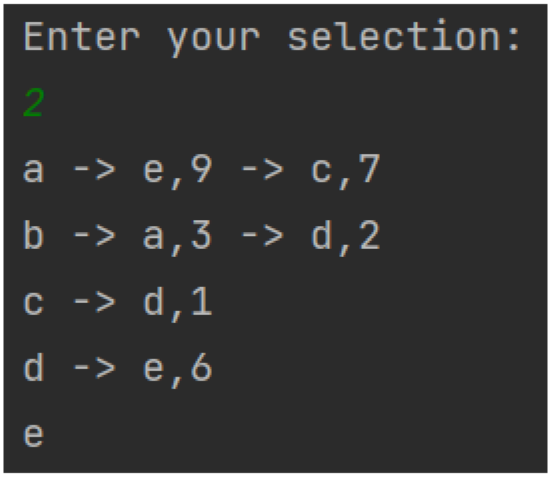
```
GraphMan main menu:
-------------------
Please select from one of the following:
[0] Exit
[1] Print Adjacency Matrix
[2] Print Adjacency List
[3] Shortest Path Between Nodes
[4] Breadth-First Search
[5] Depth-First Search
[6] Ascending Degree Depth-First Search

Enter your selection:
```

Selection option 1 (Adjacency Matrix):



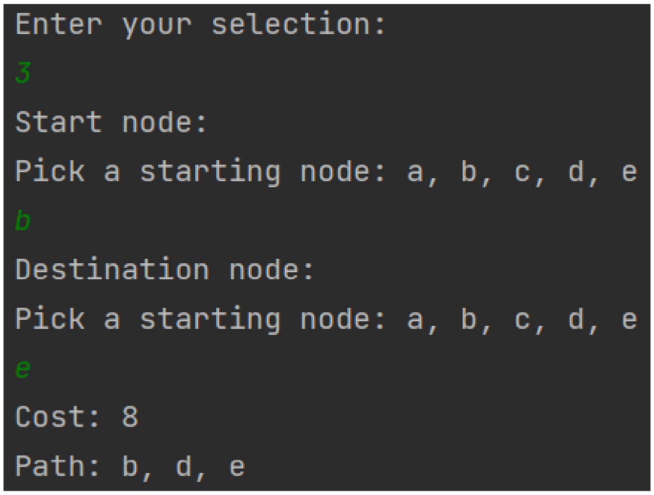
```
Enter your selection:
1
Adjacency matrix:
  a b c d e
a 0 0 1 0 1
b 1 0 0 1 0
c 0 0 0 1 0
d 0 0 0 0 1
e 0 0 0 0 0
```

Selecting option 2 (Adjacency List):



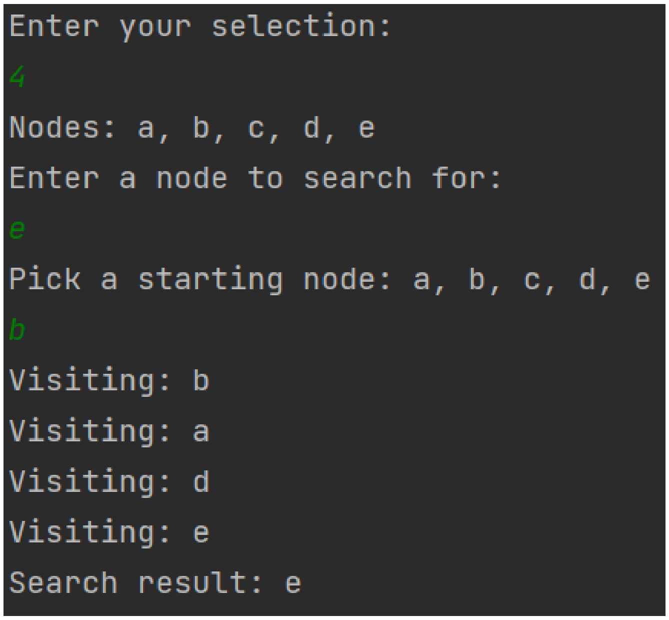
```
Enter your selection:
2
a -> e,9 -> c,7
b -> a,3 -> d,2
c -> d,1
d -> e,6
e
```

Selecting option 3 (Shortest path between 2 nodes):



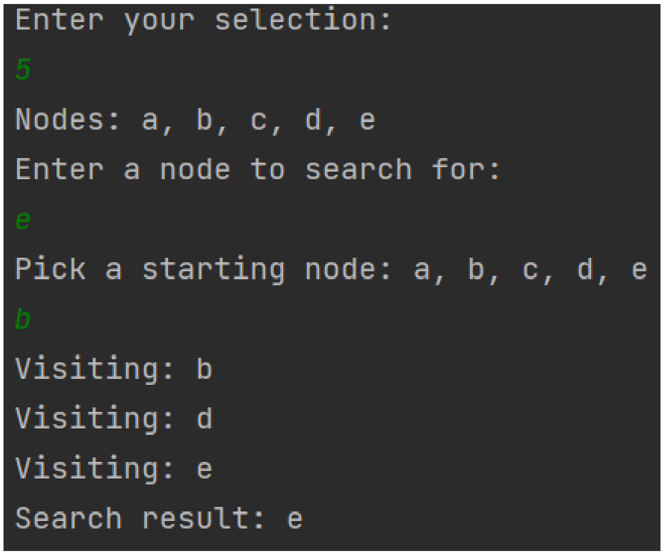
```
Enter your selection:
3
Start node:
Pick a starting node: a, b, c, d, e
b
Destination node:
Pick a starting node: a, b, c, d, e
e
Cost: 8
Path: b, d, e
```

Selecting option 4 (Breadth-first search):



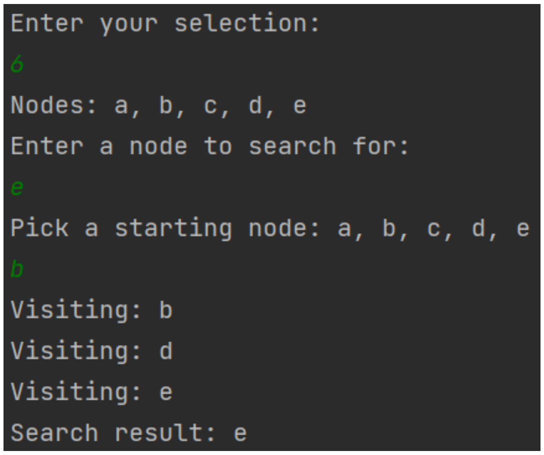
```
Enter your selection:
4
Nodes: a, b, c, d, e
Enter a node to search for:
e
Pick a starting node: a, b, c, d, e
b
Visiting: b
Visiting: a
Visiting: d
Visiting: e
Search result: e
```

Selecting option 5 (Depth-first search):



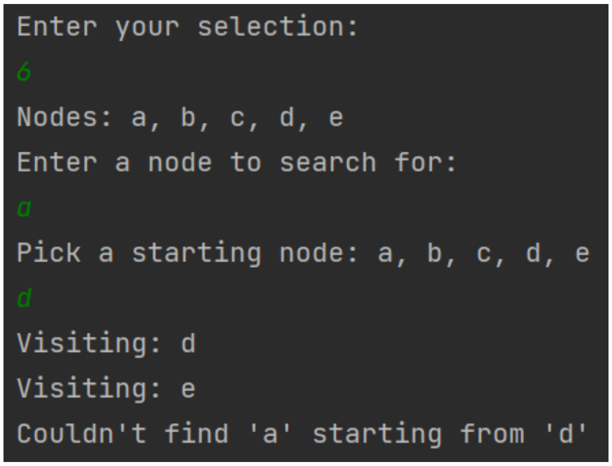
```
Enter your selection:
5
Nodes: a, b, c, d, e
Enter a node to search for:
e
Pick a starting node: a, b, c, d, e
b
Visiting: b
Visiting: d
Visiting: e
Search result: e
```

Selecting option 6 (Ordered Depth-first search):



```
Enter your selection:
6
Nodes: a, b, c, d, e
Enter a node to search for:
e
Pick a starting node: a, b, c, d, e
b
Visiting: b
Visiting: d
Visiting: e
Search result: e
```

Option 6 with no path:



```
Enter your selection:
6
Nodes: a, b, c, d, e
Enter a node to search for:
a
Pick a starting node: a, b, c, d, e
d
Visiting: d
Visiting: e
Couldn't find 'a' starting from 'd'
```