CSE 464 Software QA and Testing

Project Part 1

Name	ASU ID	ASURITE ID
Abhishek Ramakrishnamoorthy	**1225430440**	**aramak21**

The project is implemented with the help of the Java libraries such as **graphviz-java.** The dependencies are resolved by running **Maven Clean Install.** The JDK used is **Java 8.**

In addition to the libraries, user-defined classes are written to construct the graph and make the required modifications to the imported graph file.

Parsing is done using functions from the graphviz-java library.

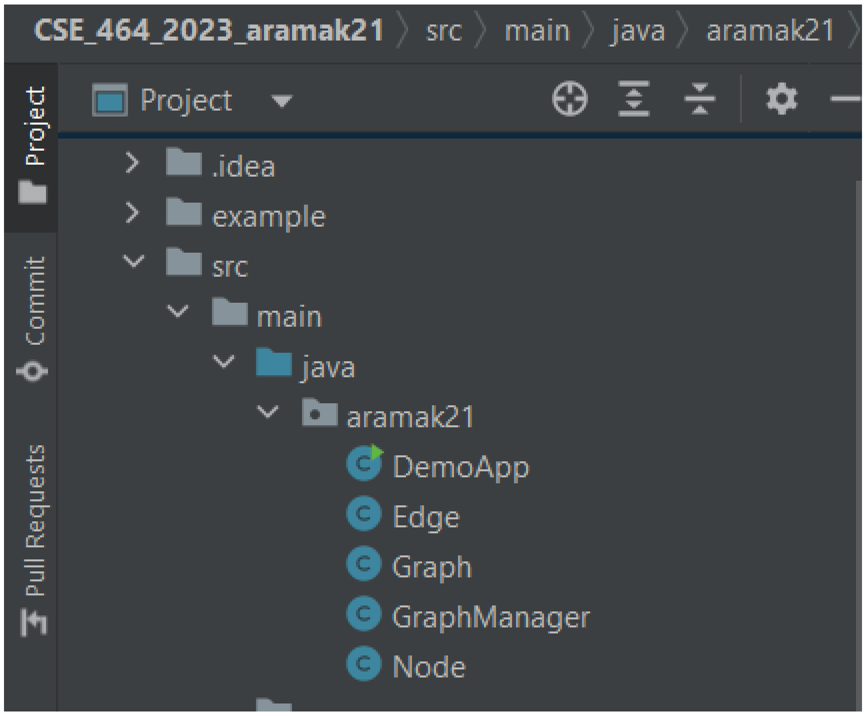
Note: It is suggested that the user sticks to either upper or lower case throughout the program for executing the various parts/features.

Note: Nodes and Edges are space sensitive. I.e node with a name “a<space>” is different from node with name “a”.

The program is not hard-coded. It is an interactive program that gets input from the user and based on the input performs the functionality.

Running the program:

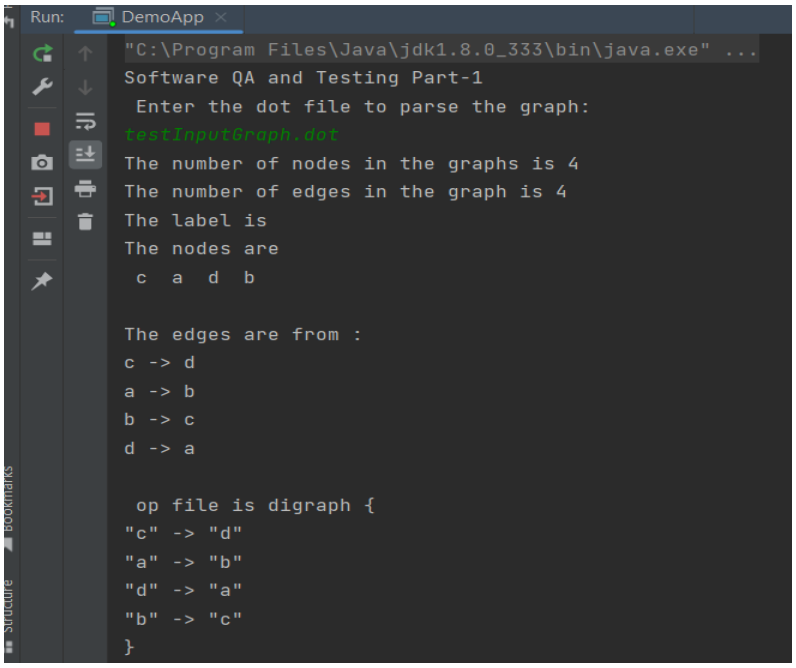
To run the program, you need to execute the **DemoApp.java file.** It can be found in **src/main/java/aramak21/DemoApp.java**



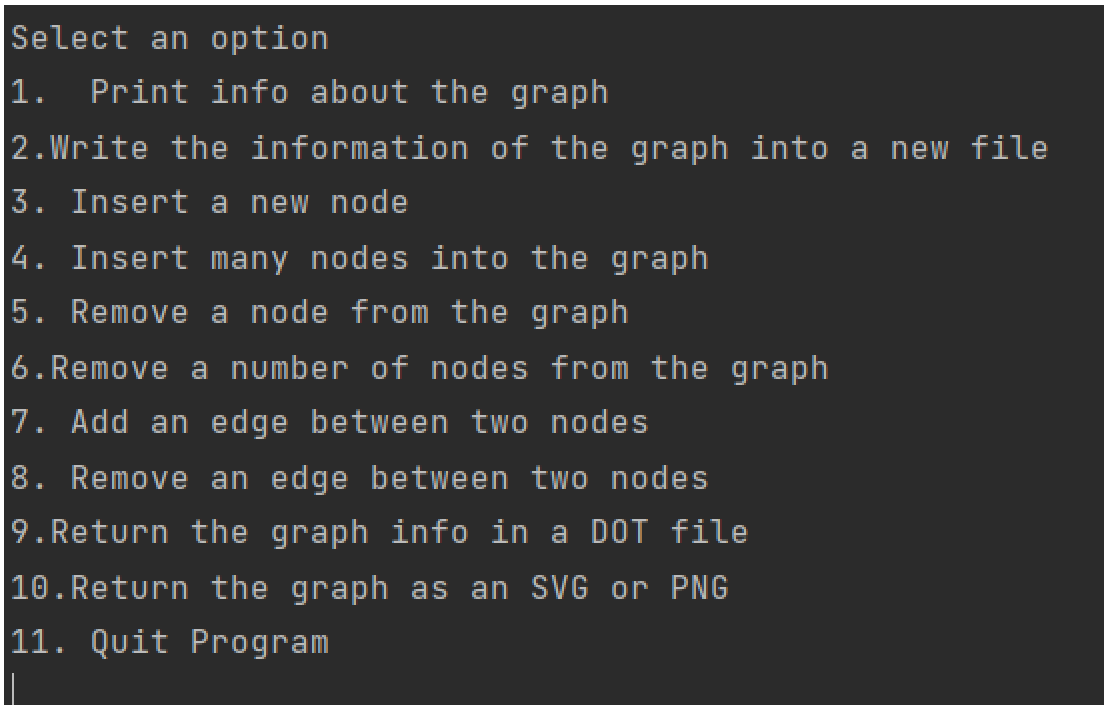
Upon running the program the user is asked to input the dot.file which is to be parsed.
[Note: it is recommended to include the "**.dot"** extension along with the filename.

Parsing the dot file. :

Upon entering a valid dot file. The dot file is parsed and the information extracted is displayed.



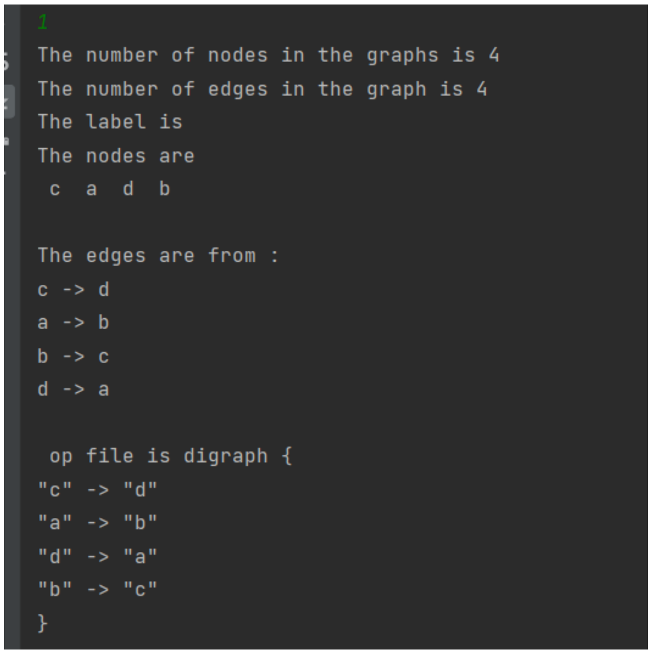
```
"C:\Program Files\Java\jdk1.8.0_333\bin\java.exe" ...
Software QA and Testing Part-1
 Enter the dot file to parse the graph:
testInputGraph.dot
The number of nodes in the graphs is 4
The number of edges in the graph is 4
The label is
The nodes are
 c  a  d  b

The edges are from :
c -> d
a -> b
b -> c
d -> a

 op file is digraph {
"c" -> "d"
"a" -> "b"
"d" -> "a"
"b" -> "c"
}
```



```
Select an option
1.  Print info about the graph
2.Write the information of the graph into a new file
3. Insert a new node
4. Insert many nodes into the graph
5. Remove a node from the graph
6.Remove a number of nodes from the graph
7. Add an edge between two nodes
8. Remove an edge between two nodes
9.Return the graph info in a DOT file
10.Return the graph as an SVG or PNG
11. Quit Program
```

Option 1 → API for printing a graph: toString()

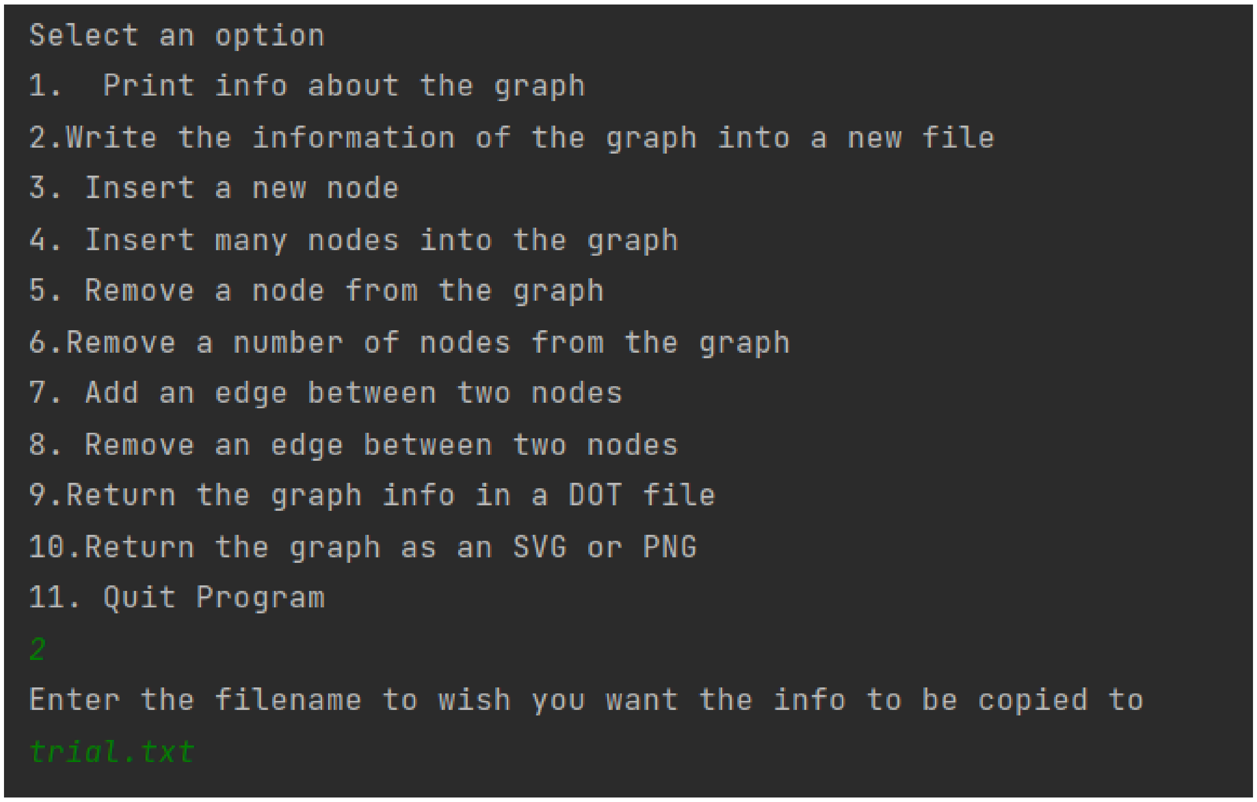
Here we have implemented Feature 1 where we parse the graph, and upon request, we display all information about the graph such as the number of nodes,edges, the labels of the nodes and edges.



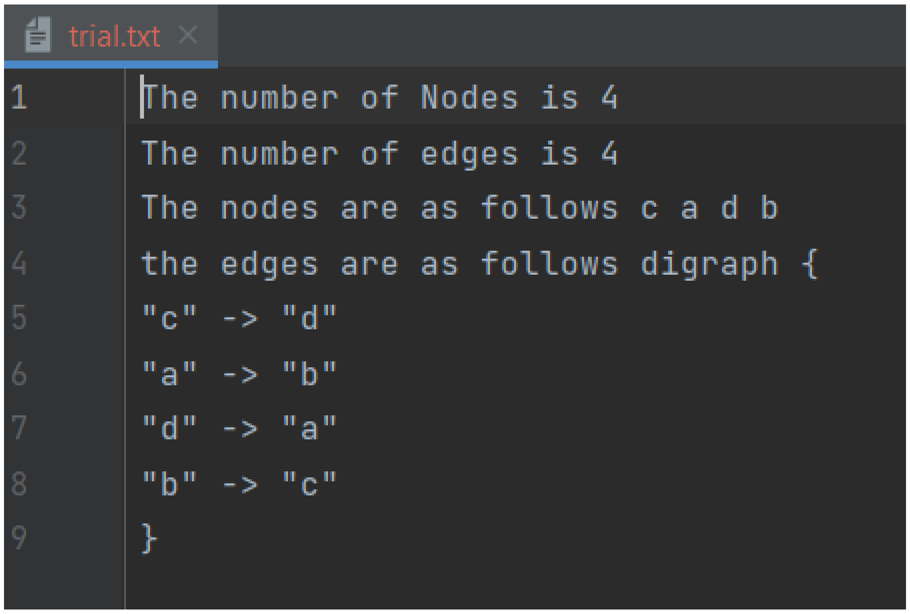
```
1
The number of nodes in the graphs is 4
The number of edges in the graph is 4
The label is
The nodes are
 c  a  d  b

The edges are from :
c -> d
a -> b
b -> c
d -> a

 op file is digraph {
"c" -> "d"
"a" -> "b"
"d" -> "a"
"b" -> "c"
}
```

<u>Option 2 → API for output to file: outputGraph(String filepath)</u>

The user is asked to enter the filepath or filename. A new file is created and the contents of the graph will be written into the filename provided by the user.



```
Select an option
1.  Print info about the graph
2.Write the information of the graph into a new file
3. Insert a new node
4. Insert many nodes into the graph
5. Remove a node from the graph
6.Remove a number of nodes from the graph
7. Add an edge between two nodes
8. Remove an edge between two nodes
9.Return the graph info in a DOT file
10.Return the graph as an SVG or PNG
11. Quit Program
2
Enter the filename to wish you want the info to be copied to
trial.txt
```

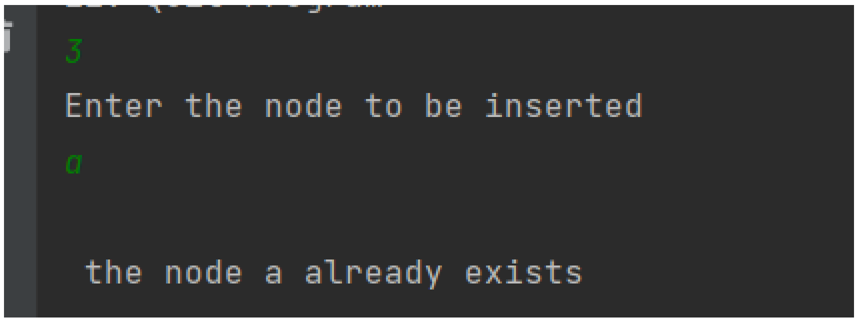


```
The number of Nodes is 4
The number of edges is 4
The nodes are as follows c a d b
the edges are as follows digraph {
"c" -> "d"
"a" -> "b"
"d" -> "a"
"b" -> "c"
}
```

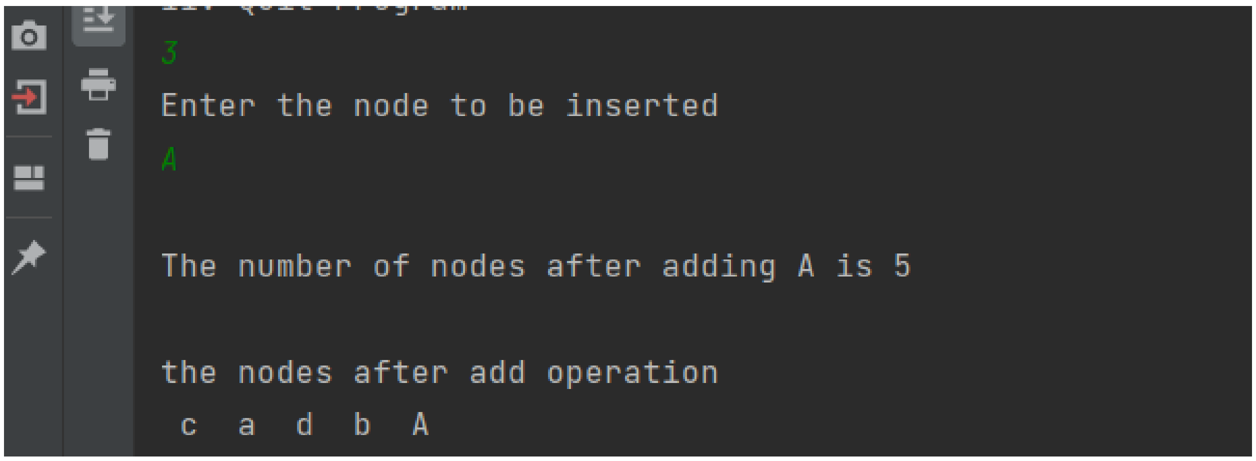
Option 3 →API to Add a node and check of duplicate labels: addNode(String label)

The user is asked to input the node which is to be inserted. If the node already exists in the graph, it displays the same as a message. Otherwise, the new node is created and inserted into the graph.

Note : **A** and **a** are not the same nodes.



```
3
Enter the node to be inserted
a

 the node a already exists
```

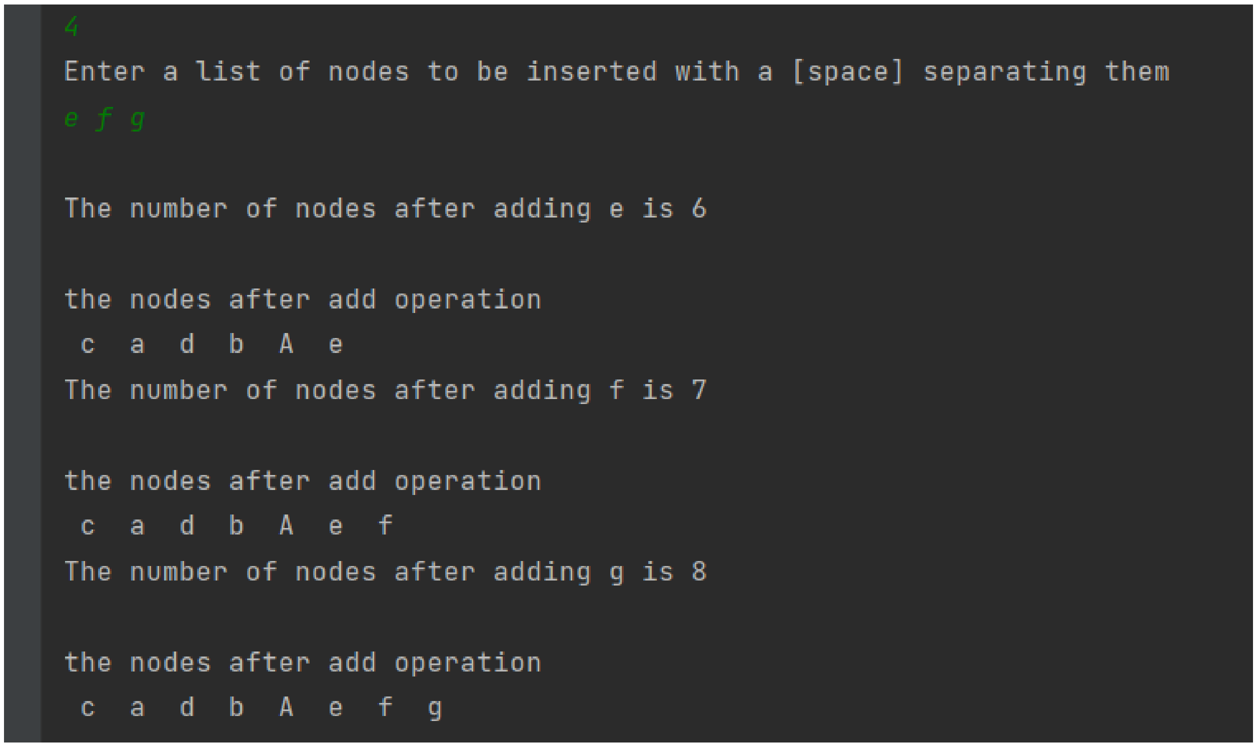


```
3
Enter the node to be inserted
A

The number of nodes after adding A is 5

the nodes after add operation
 c  a  d  b  A
```

Option 4 : API to Add a list of nodes: addNodes(String[] label)

Here the user is asked to input the list of nodes separated by a space. Note : <Space> is used as the delimiter and accordingly the nodes are created.

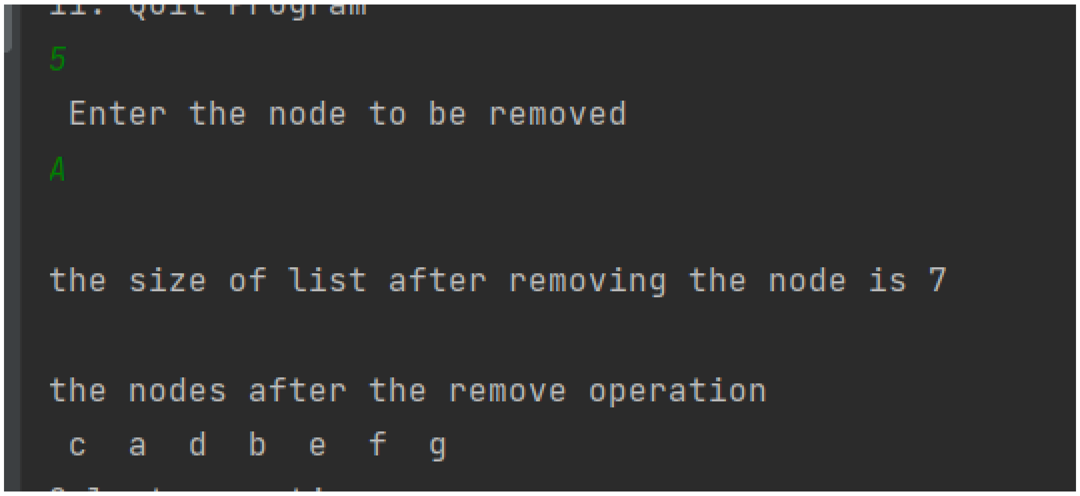
Similar to how a ndoe is inserted, here the logic checks if the node is already present, and if not, it creates a new node.



```
4
Enter a list of nodes to be inserted with a [space] separating them
e f g

The number of nodes after adding e is 6

the nodes after add operation
 c  a  d  b  A  e
The number of nodes after adding f is 7

the nodes after add operation
 c  a  d  b  A  e  f
The number of nodes after adding g is 8

the nodes after add operation
 c  a  d  b  A  e  f  g
```

Option 5 → Remove a node: removeNode(String label)

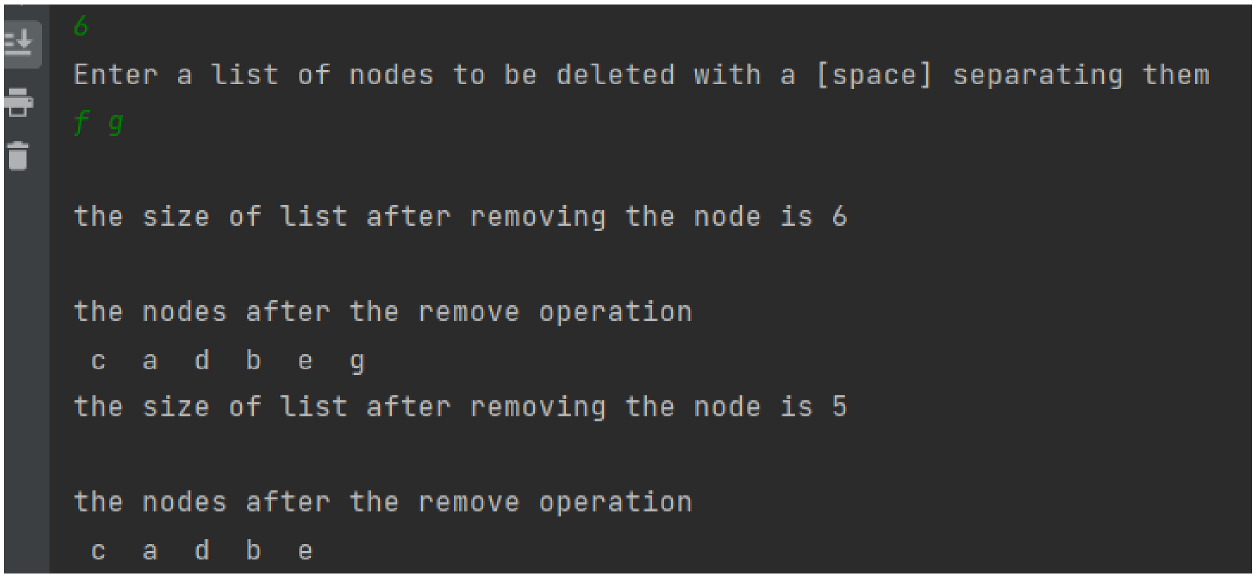
The user is asked to input the name of node which is to be removed. If no such node exists nothing happens. If the node exists in the graph, then it is removed. Removing the node also removes the edges between the connected nodes.



```
11. Quit Program
5
 Enter the node to be removed
A


the size of list after removing the node is 7


the nodes after the remove operation
 c  a  d  b  e  f  g
```

Option 6 →> API to Remove a list of nodes: removeNodes(String[] label)

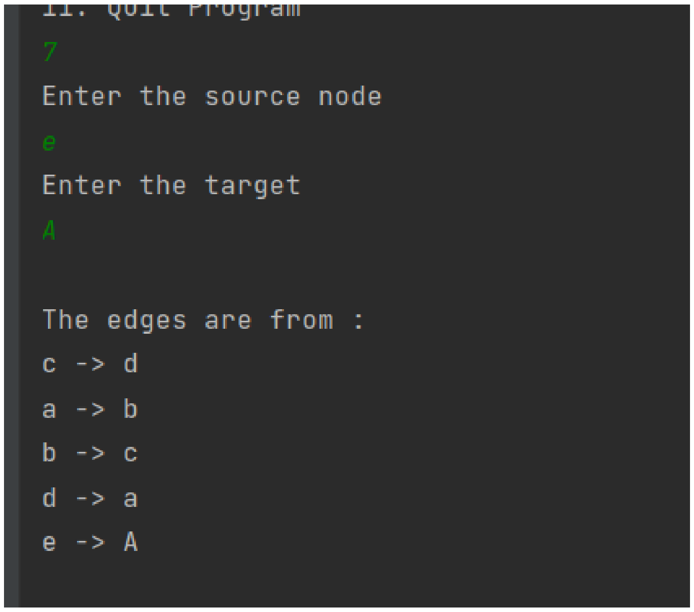
Here the user is asked to input the list of nodes separated by a space. Note : <Space> is used as the delimiter and accordingly the nodes are removed. If no such node exists nothing happens. If the node exists in the graph, then it is removed.



```
6
Enter a list of nodes to be deleted with a [space] separating them
f g

the size of list after removing the node is 6

the nodes after the remove operation
 c  a  d  b  e  g
the size of list after removing the node is 5

the nodes after the remove operation
 c  a  d  b  e
```

Option 7 → API to Add an edge and check of duplicate edges: addEdge(String srcLabel, String dstLabel)

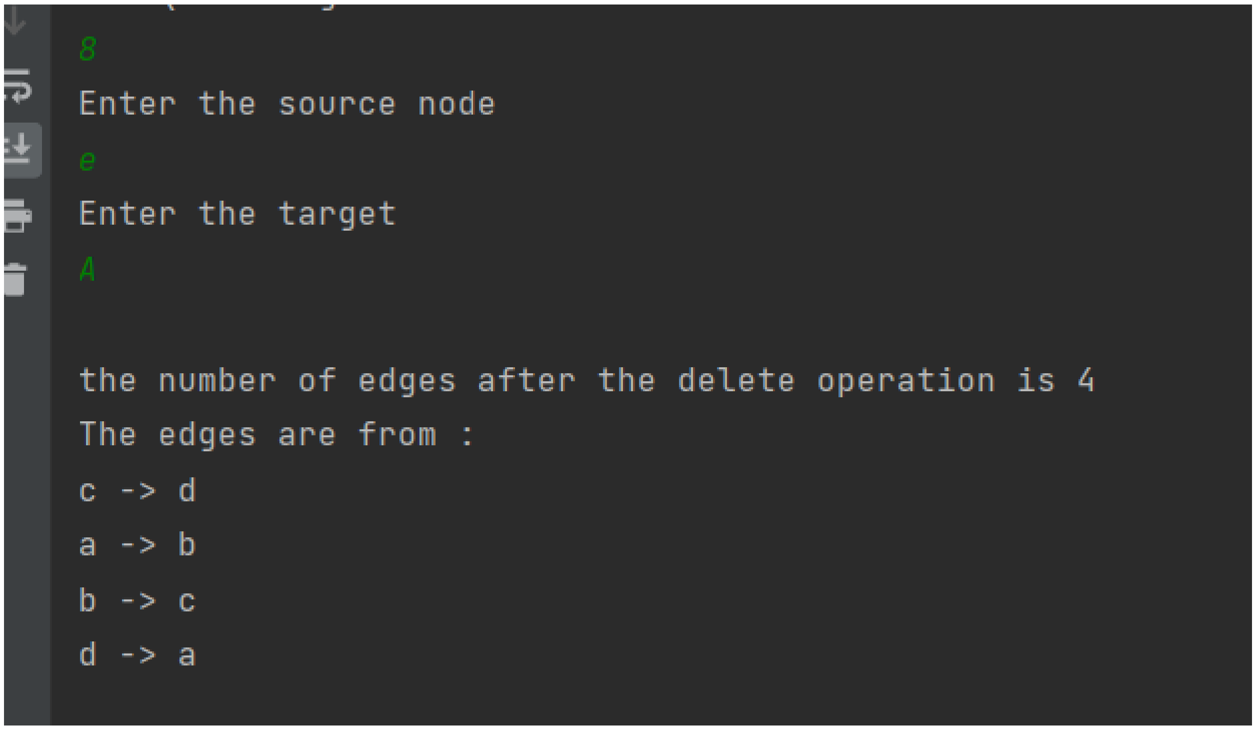
The user is asked to input the source and destination nodes. If both the nodes already exist, then an edge is created. If one among the source or destination is not present, a new node is created and the edge is added.



```
11. Quit Program
7
Enter the source node
e
Enter the target
A

The edges are from :
c -> d
a -> b
b -> c
d -> a
e -> A
```

Option 8 → API to Remove an edge: removeEdge(String srcLabel, String dstLabel)

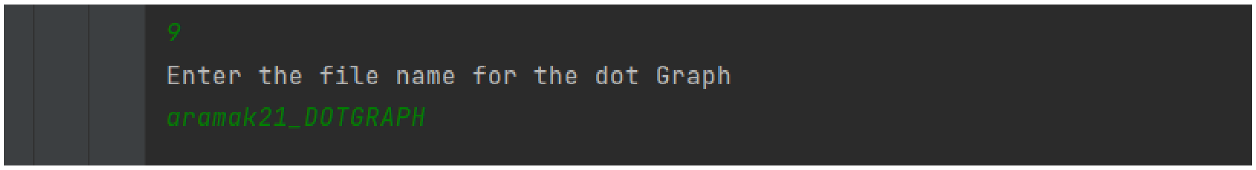
The user is asked to source and destination nodes between which the edge is to be removed. If no such edge exists, then the program does nothing. If there is a valid edge between the nodes, then it is removed from the graph.



```
8
Enter the source node
e
Enter the target
A

the number of edges after the delete operation is 4
The edges are from :
c -> d
a -> b
b -> c
d -> a
```

Option 9 → API to output the imported graph into a DOT file:

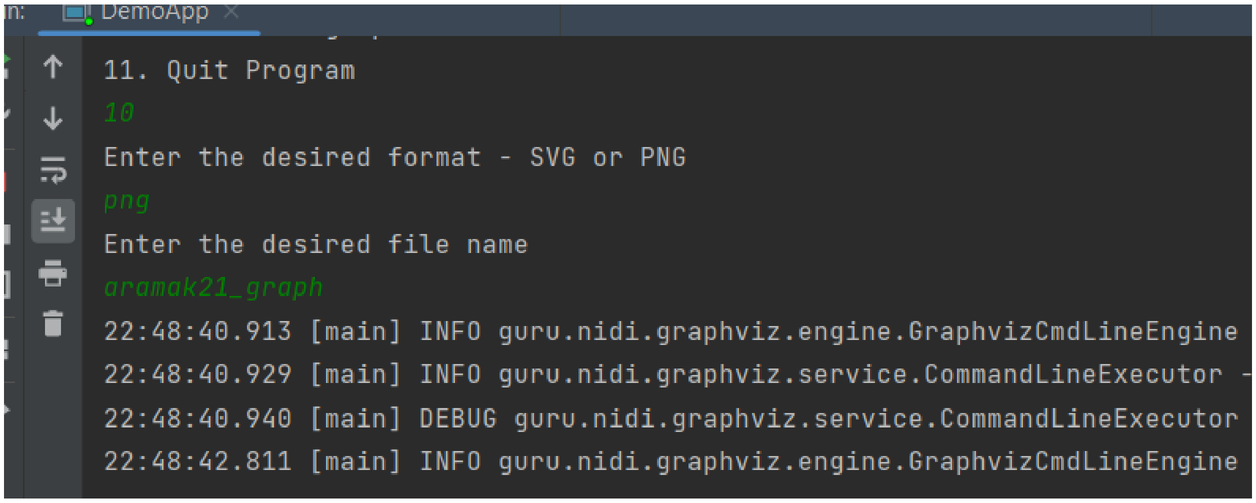
The user is asked to input the name of the dotfile to which the user desires the graph to be imported to.



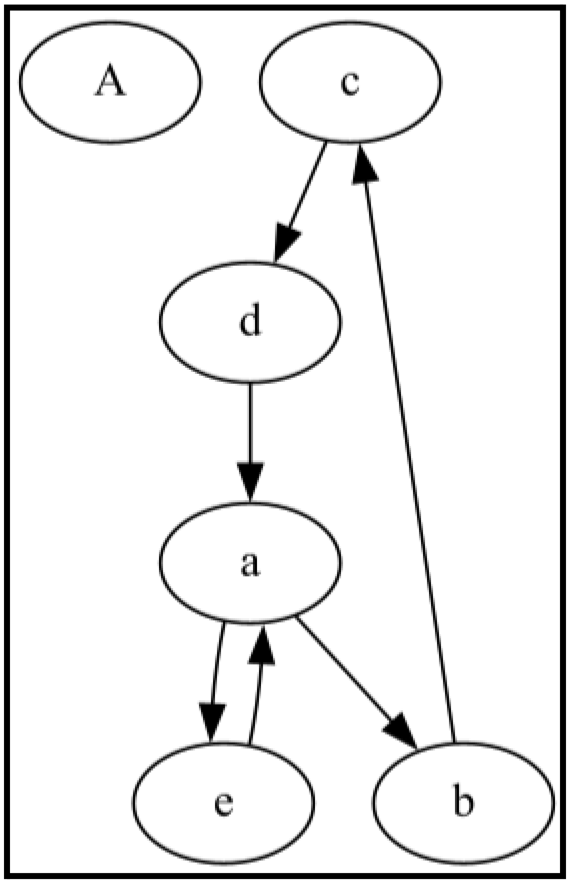
```
9
Enter the file name for the dot Graph
aramak21_DOTGRAPH
```

Option 10 → API: outputGraphics(String path, String format)

The user is asked to input the desired format [svg/png] and the desired file name. The graph is imported into the desired format.



```
11. Quit Program
10
Enter the desired format - SVG or PNG
png
Enter the desired file name
aramak21_graph
22:48:40.913 [main] INFO guru.nidi.graphviz.engine.GraphvizCmdLineEngine
22:48:40.929 [main] INFO guru.nidi.graphviz.service.CommandLineExecutor -
22:48:40.940 [main] DEBUG guru.nidi.graphviz.service.CommandLineExecutor
22:48:42.811 [main] INFO guru.nidi.graphviz.engine.GraphvizCmdLineEngine
```



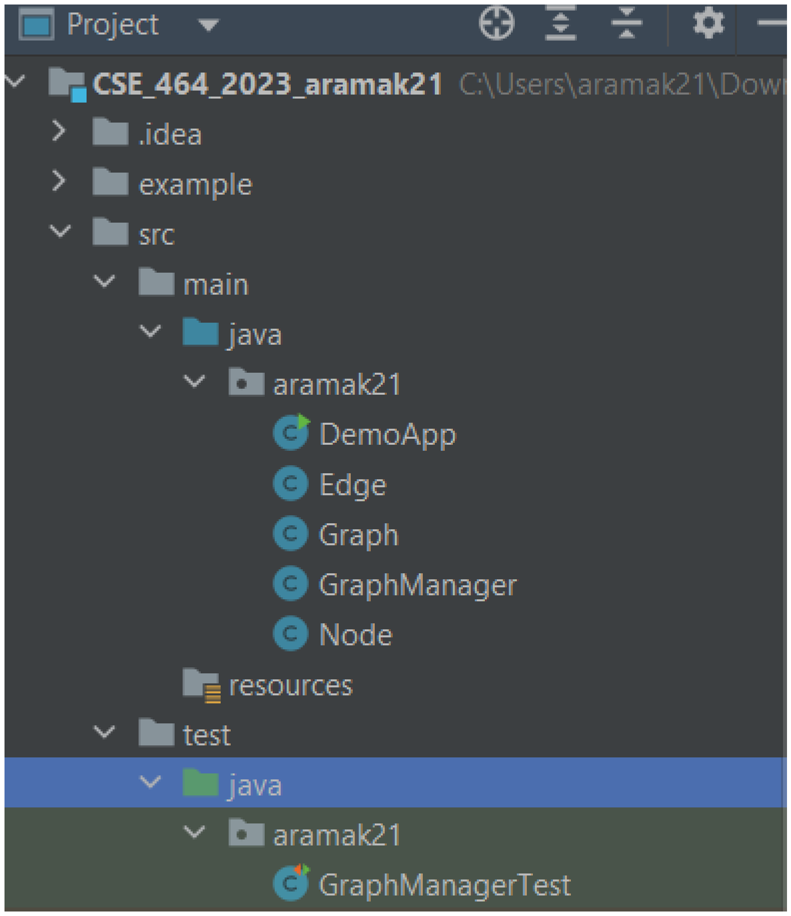
Option 11 → Quits the program:

<u>Testing the features:</u>

The test file is located in the directory ***src/test/java/aramak21/GraphManagerTest.java***
The GraphManagerTest.java has unit test cases written for all the features which were implemented in the program.

Upon executing the file, the tests for the various features or APIs are run.

Note Tests can be executed in IntelliJ or Maven Test.


Project
CSE_464_2023_aramak21 C:\Users\aramak21\Dow
.idea
example
src
main
java
aramak21
DemoApp
Edge
Graph
GraphManager
Node
resources
test
java
aramak21
GraphManagerTest