Homework OF 31-03-2025

Subject: CSW2 (CSE 2141)

Name: Arpit Kumar Mohanty

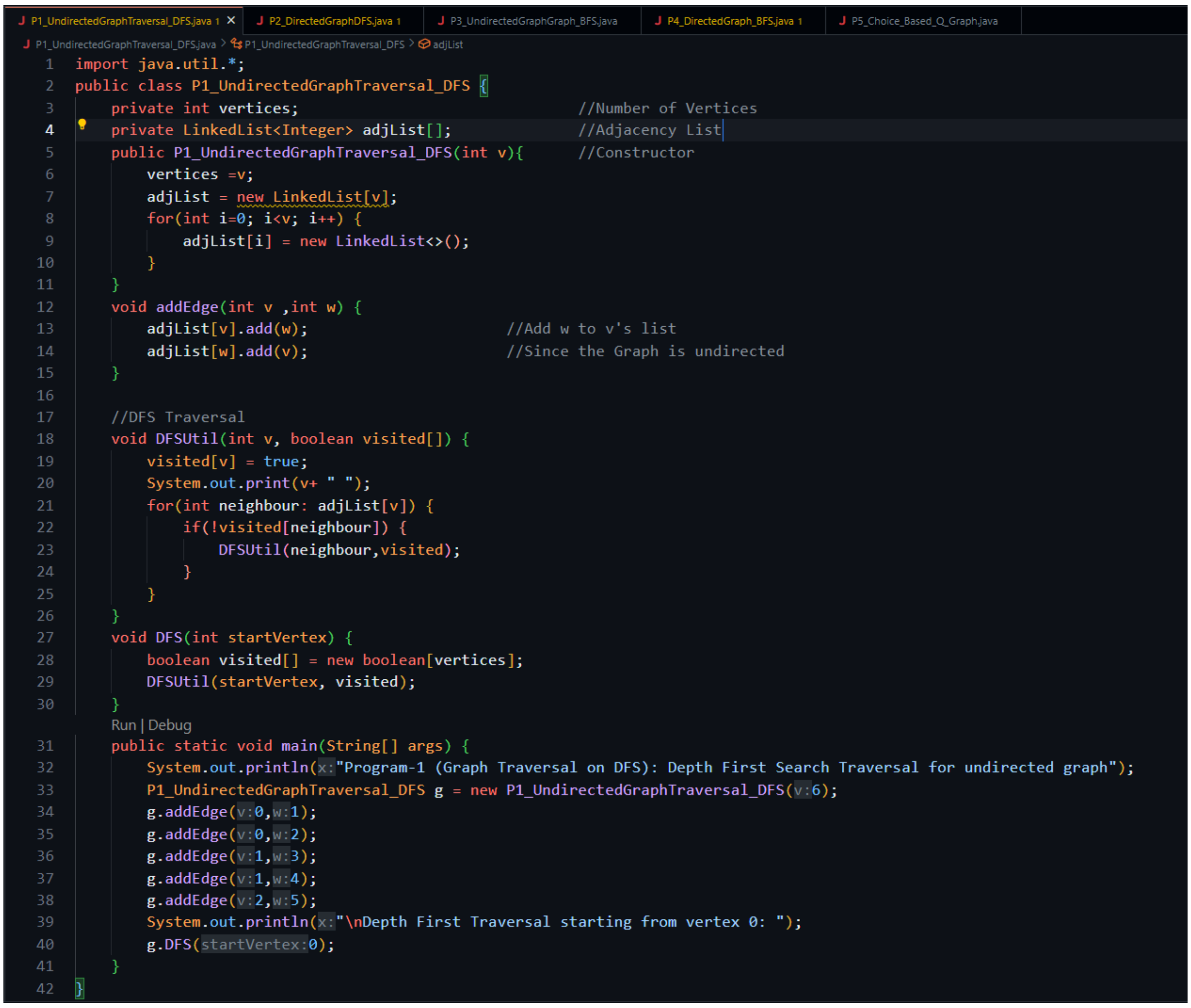
Registration Number: 2341013237

Section: 23412G1

Branch: CSE

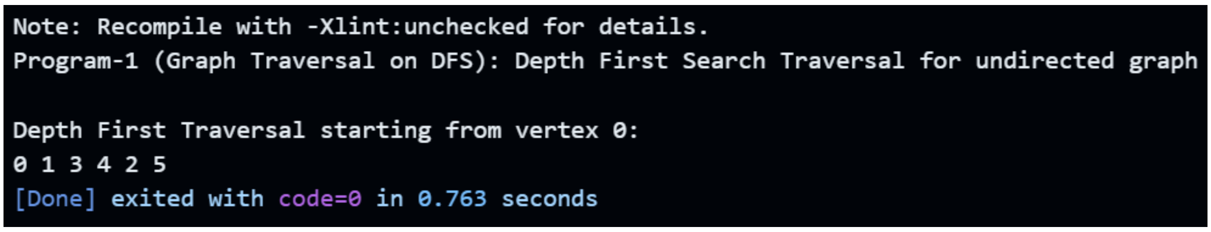
Q1. Write Java program on Depth First Search Traversal for undirected graph

Solution:


P1_UndirectedGraphTraversal_DFS.java 1
P2_DirectedGraphDFS.java 1
P3_UndirectedGraphGraph_BFS.java
P4_DirectedGraph_BFS.java 1
P5_Choice_Based_Q_Graph.java
P1_UndirectedGraphTraversal_DFS.java > P1_UndirectedGraphTraversal_DFS > adjList
Run | Debug

```java
import java.util.*;
public class P1_UndirectedGraphTraversal_DFS {
    private int vertices;                                   //Number of Vertices
    private LinkedList<Integer> adjList[];                  //Adjacency List
    public P1_UndirectedGraphTraversal_DFS(int v){          //Constructor
        vertices =v;
        adjList = new LinkedList[v];
        for(int i=0; i<v; i++) {
            adjList[i] = new LinkedList<>();
        }
    }
    void addEdge(int v ,int w) {
        adjList[v].add(w);                          //Add w to v's list
        adjList[w].add(v);                          //Since the Graph is undirected
    }

    //DFS Traversal
    void DFSUtil(int v, boolean visited[]) {
        visited[v] = true;
        System.out.print(v+ " ");
        for(int neighbour: adjList[v]) {
            if(!visited[neighbour]) {
                DFSUtil(neighbour,visited);
            }
        }
    }
    void DFS(int startVertex) {
        boolean visited[] = new boolean[vertices];
        DFSUtil(startVertex, visited);
    }
    public static void main(String[] args) {
        System.out.println(x:"Program-1 (Graph Traversal on DFS): Depth First Search Traversal for undirected graph");
        P1_UndirectedGraphTraversal_DFS g = new P1_UndirectedGraphTraversal_DFS(v:6);
        g.addEdge(v:0,w:1);
        g.addEdge(v:0,w:2);
        g.addEdge(v:1,w:3);
        g.addEdge(v:1,w:4);
        g.addEdge(v:2,w:5);
        System.out.println(x:"\nDepth First Traversal starting from vertex 0: ");
        g.DFS(startVertex:0);
    }
}
```

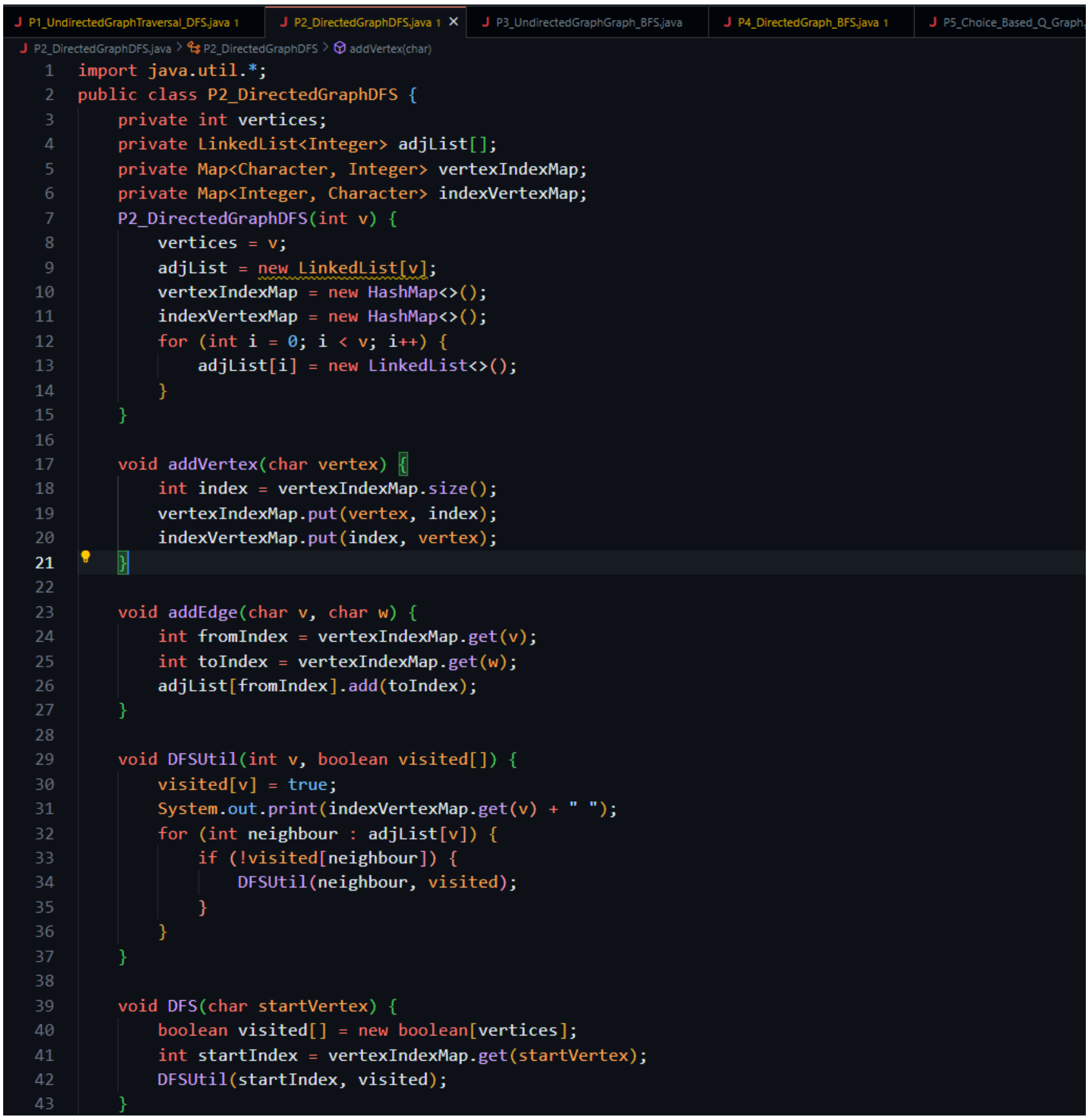
Output:



```
Note: Recompile with -Xlint:unchecked for details.
Program-1 (Graph Traversal on DFS): Depth First Search Traversal for undirected graph

Depth First Traversal starting from vertex 0:
0 1 3 4 2 5
[Done] exited with code=0 in 0.763 seconds
```

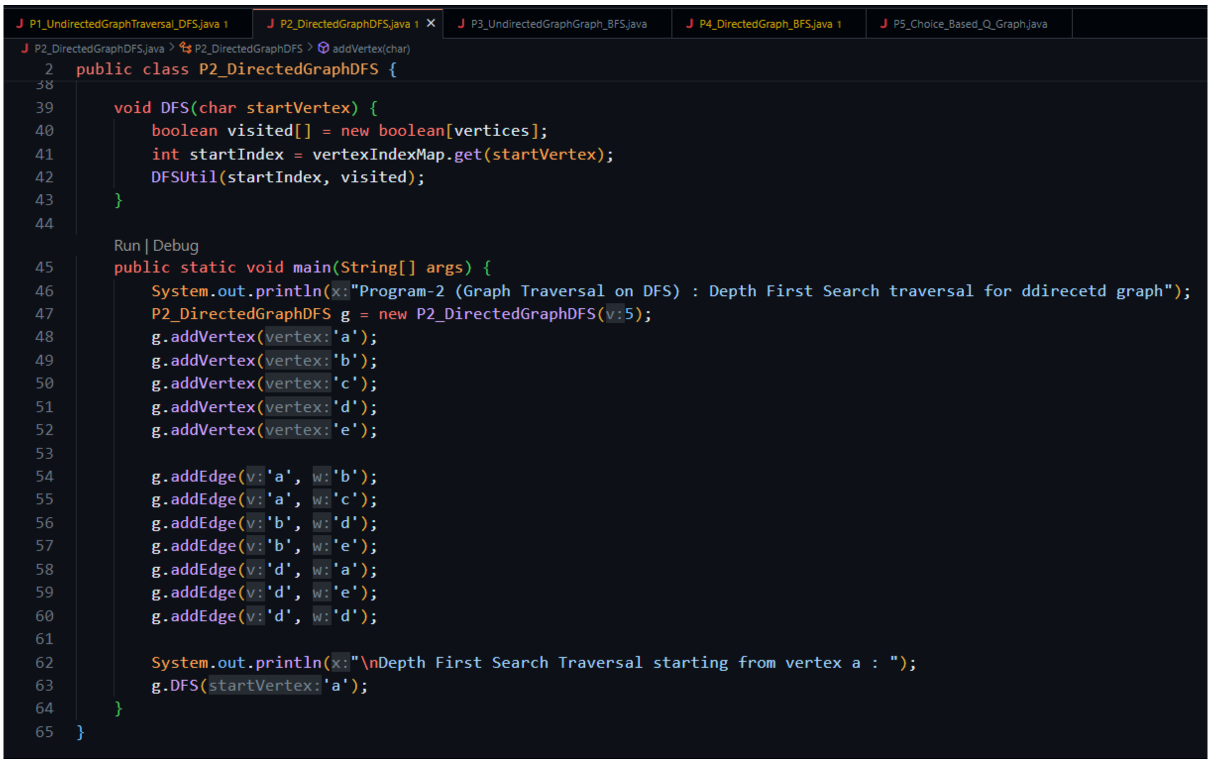
Q2. Write Java program on Depth First Search Traversal for directed graph

Solution:



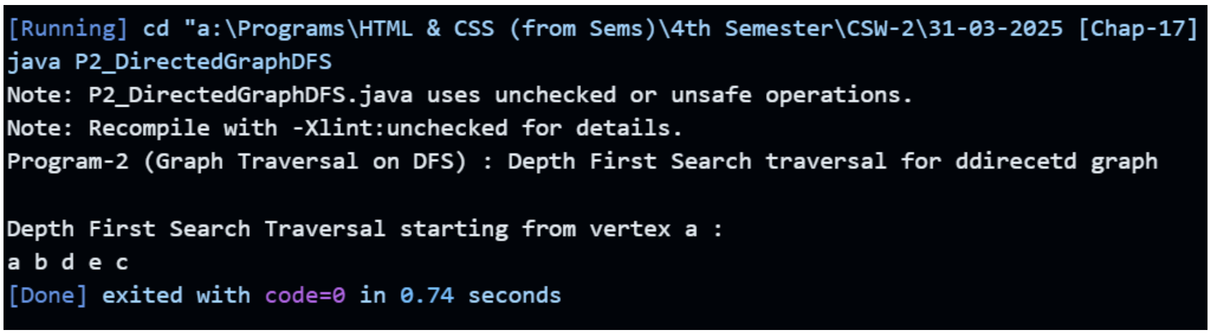
P1_UndirectedGraphTraversal_DFS.java 1
P2_DirectedGraphDFS.java 1
P3_UndirectedGraphGraph_BFS.java
P4_DirectedGraph_BFS.java 1
P5_Choice_Based_Q_Graph
P2_DirectedGraphDFS.java > P2_DirectedGraphDFS > addVertex(char)

```java
import java.util.*;
public class P2_DirectedGraphDFS {
    private int vertices;
    private LinkedList<Integer> adjList[];
    private Map<Character, Integer> vertexIndexMap;
    private Map<Integer, Character> indexVertexMap;
    P2_DirectedGraphDFS(int v) {
        vertices = v;
        adjList = new LinkedList[v];
        vertexIndexMap = new HashMap<>();
        indexVertexMap = new HashMap<>();
        for (int i = 0; i < v; i++) {
            adjList[i] = new LinkedList<>();
        }
    }

    void addVertex(char vertex) {
        int index = vertexIndexMap.size();
        vertexIndexMap.put(vertex, index);
        indexVertexMap.put(index, vertex);
    }

    void addEdge(char v, char w) {
        int fromIndex = vertexIndexMap.get(v);
        int toIndex = vertexIndexMap.get(w);
        adjList[fromIndex].add(toIndex);
    }

    void DFSUtil(int v, boolean visited[]) {
        visited[v] = true;
        System.out.print(indexVertexMap.get(v) + " ");
        for (int neighbour : adjList[v]) {
            if (!visited[neighbour]) {
                DFSUtil(neighbour, visited);
            }
        }
    }

    void DFS(char startVertex) {
        boolean visited[] = new boolean[vertices];
        int startIndex = vertexIndexMap.get(startVertex);
        DFSUtil(startIndex, visited);
    }
```


P1_UndirectedGraphTraversal_DFS.java 1
P2_DirectedGraphDFS.java 1
P3_UndirectedGraphGraph_BFS.java
P4_DirectedGraph_BFS.java 1
P5_Choice_Based_Q_Graph.java
P2_DirectedGraphDFS.java > P2_DirectedGraphDFS > addVertex(char)
Run | Debug

```java
public class P2_DirectedGraphDFS {

    void DFS(char startVertex) {
        boolean visited[] = new boolean[vertices];
        int startIndex = vertexIndexMap.get(startVertex);
        DFSUtil(startIndex, visited);
    }

    public static void main(String[] args) {
        System.out.println(x:"Program-2 (Graph Traversal on DFS) : Depth First Search traversal for ddirecetd graph");
        P2_DirectedGraphDFS g = new P2_DirectedGraphDFS(v:5);
        g.addVertex(vertex:'a');
        g.addVertex(vertex:'b');
        g.addVertex(vertex:'c');
        g.addVertex(vertex:'d');
        g.addVertex(vertex:'e');

        g.addEdge(v:'a', w:'b');
        g.addEdge(v:'a', w:'c');
        g.addEdge(v:'b', w:'d');
        g.addEdge(v:'b', w:'e');
        g.addEdge(v:'d', w:'a');
        g.addEdge(v:'d', w:'e');
        g.addEdge(v:'d', w:'d');

        System.out.println(x:"\nDepth First Search Traversal starting from vertex a : ");
        g.DFS(startVertex:'a');
    }
}
```

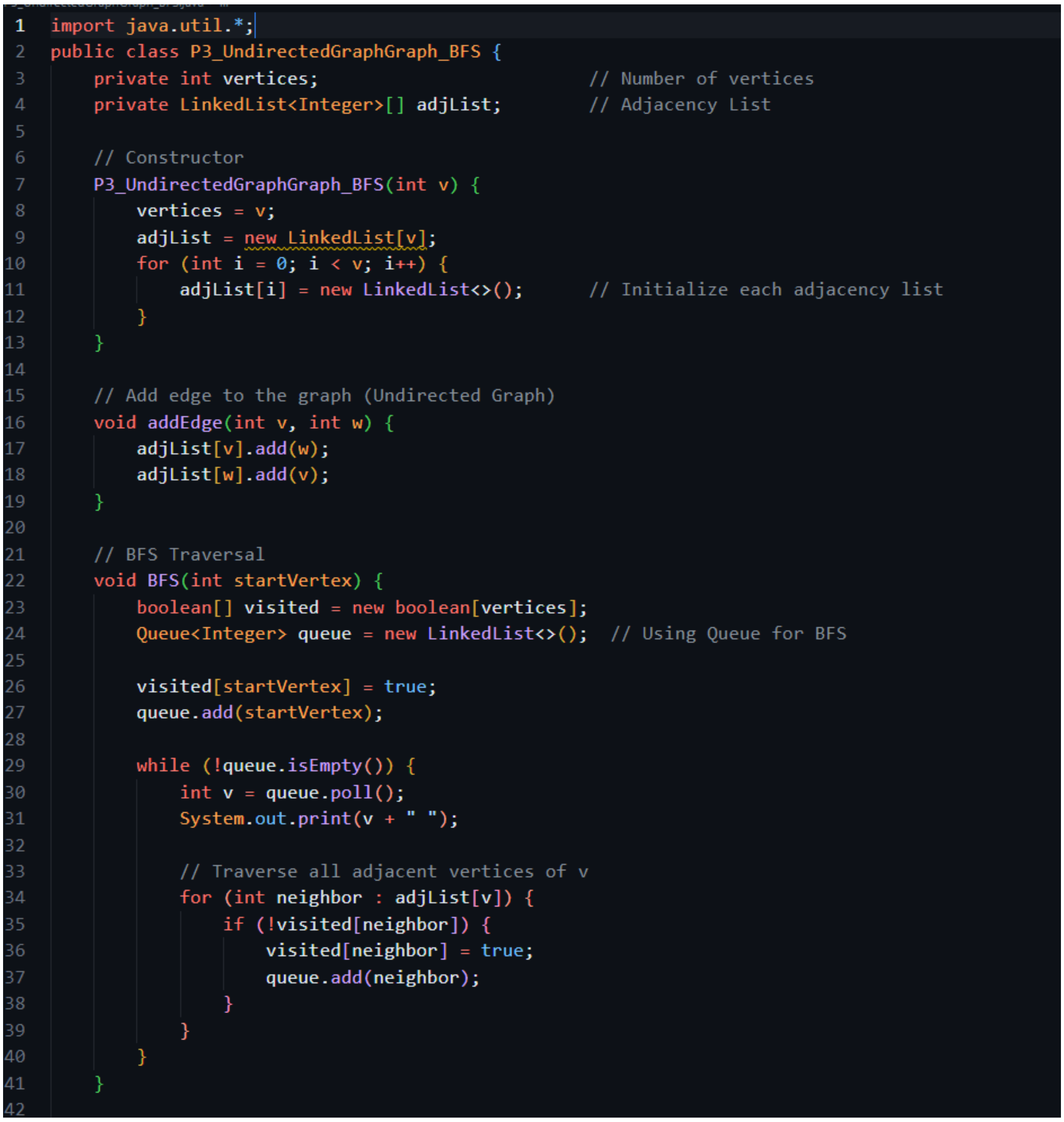
Output:

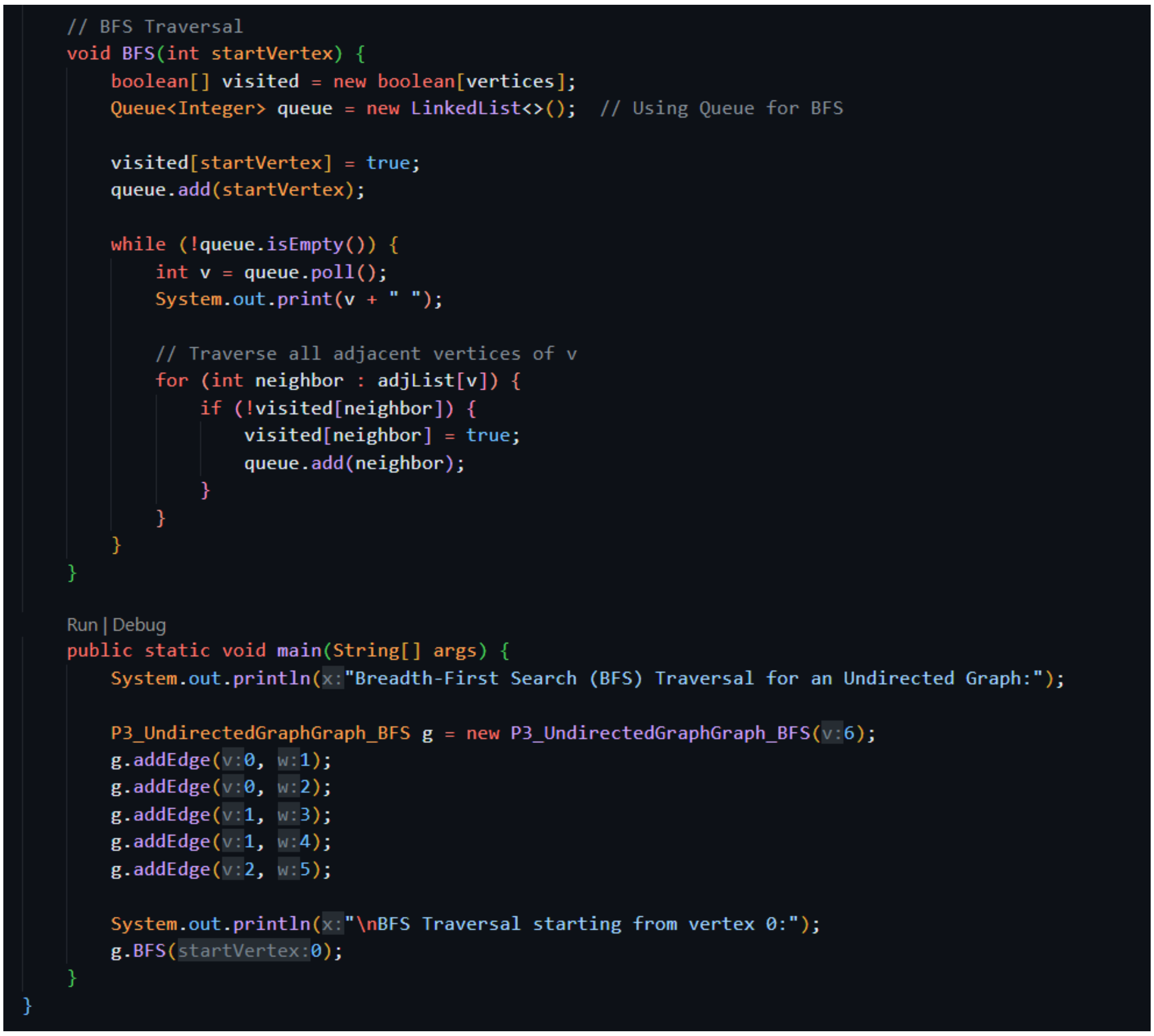


```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\31-03-2025 [Chap-17]
java P2_DirectedGraphDFS
Note: P2_DirectedGraphDFS.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Program-2 (Graph Traversal on DFS) : Depth First Search traversal for ddirecetd graph

Depth First Search Traversal starting from vertex a :
a b d e c
[Done] exited with code=0 in 0.74 seconds
```

Q3. Write Java program on Breadth First Search Traversal for undirected graph.

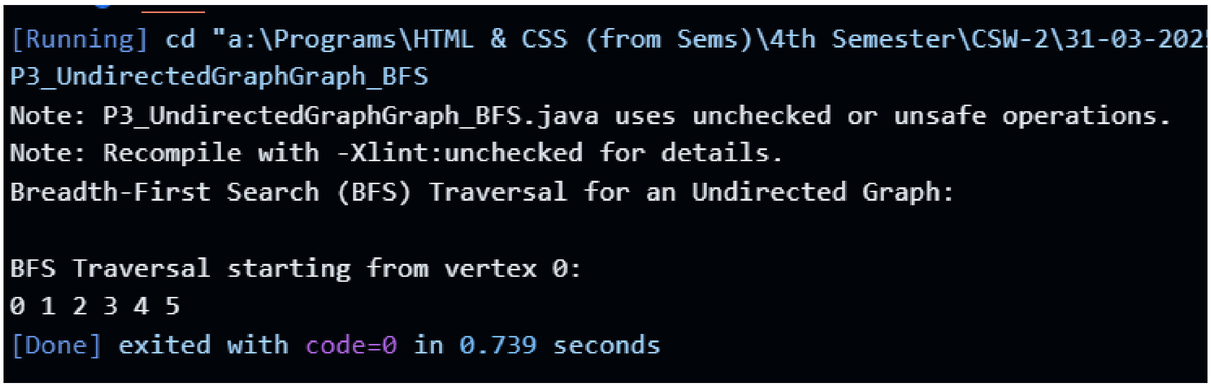
Solution:


```java
import java.util.*;
public class P3_UndirectedGraphGraph_BFS {
    private int vertices;                          // Number of vertices
    private LinkedList<Integer>[] adjList;         // Adjacency List

    // Constructor
    P3_UndirectedGraphGraph_BFS(int v) {
        vertices = v;
        adjList = new LinkedList[v];
        for (int i = 0; i < v; i++) {
            adjList[i] = new LinkedList<>();       // Initialize each adjacency list
        }
    }

    // Add edge to the graph (Undirected Graph)
    void addEdge(int v, int w) {
        adjList[v].add(w);
        adjList[w].add(v);
    }

    // BFS Traversal
    void BFS(int startVertex) {
        boolean[] visited = new boolean[vertices];
        Queue<Integer> queue = new LinkedList<>();  // Using Queue for BFS

        visited[startVertex] = true;
        queue.add(startVertex);

        while (!queue.isEmpty()) {
            int v = queue.poll();
            System.out.print(v + " ");

            // Traverse all adjacent vertices of v
            for (int neighbor : adjList[v]) {
                if (!visited[neighbor]) {
                    visited[neighbor] = true;
                    queue.add(neighbor);
                }
            }
        }
    }

```



```
    // BFS Traversal
    void BFS(int startVertex) {
        boolean[] visited = new boolean[vertices];
        Queue<Integer> queue = new LinkedList<>();  // Using Queue for BFS

        visited[startVertex] = true;
        queue.add(startVertex);

        while (!queue.isEmpty()) {
            int v = queue.poll();
            System.out.print(v + " ");

            // Traverse all adjacent vertices of v
            for (int neighbor : adjList[v]) {
                if (!visited[neighbor]) {
                    visited[neighbor] = true;
                    queue.add(neighbor);
                }
            }
        }
    }

    Run | Debug
    public static void main(String[] args) {
        System.out.println(x:"Breadth-First Search (BFS) Traversal for an Undirected Graph:");

        P3_UndirectedGraphGraph_BFS g = new P3_UndirectedGraphGraph_BFS(v:6);
        g.addEdge(v:0, w:1);
        g.addEdge(v:0, w:2);
        g.addEdge(v:1, w:3);
        g.addEdge(v:1, w:4);
        g.addEdge(v:2, w:5);

        System.out.println(x:"\nBFS Traversal starting from vertex 0:");
        g.BFS(startVertex:0);
    }
}
```

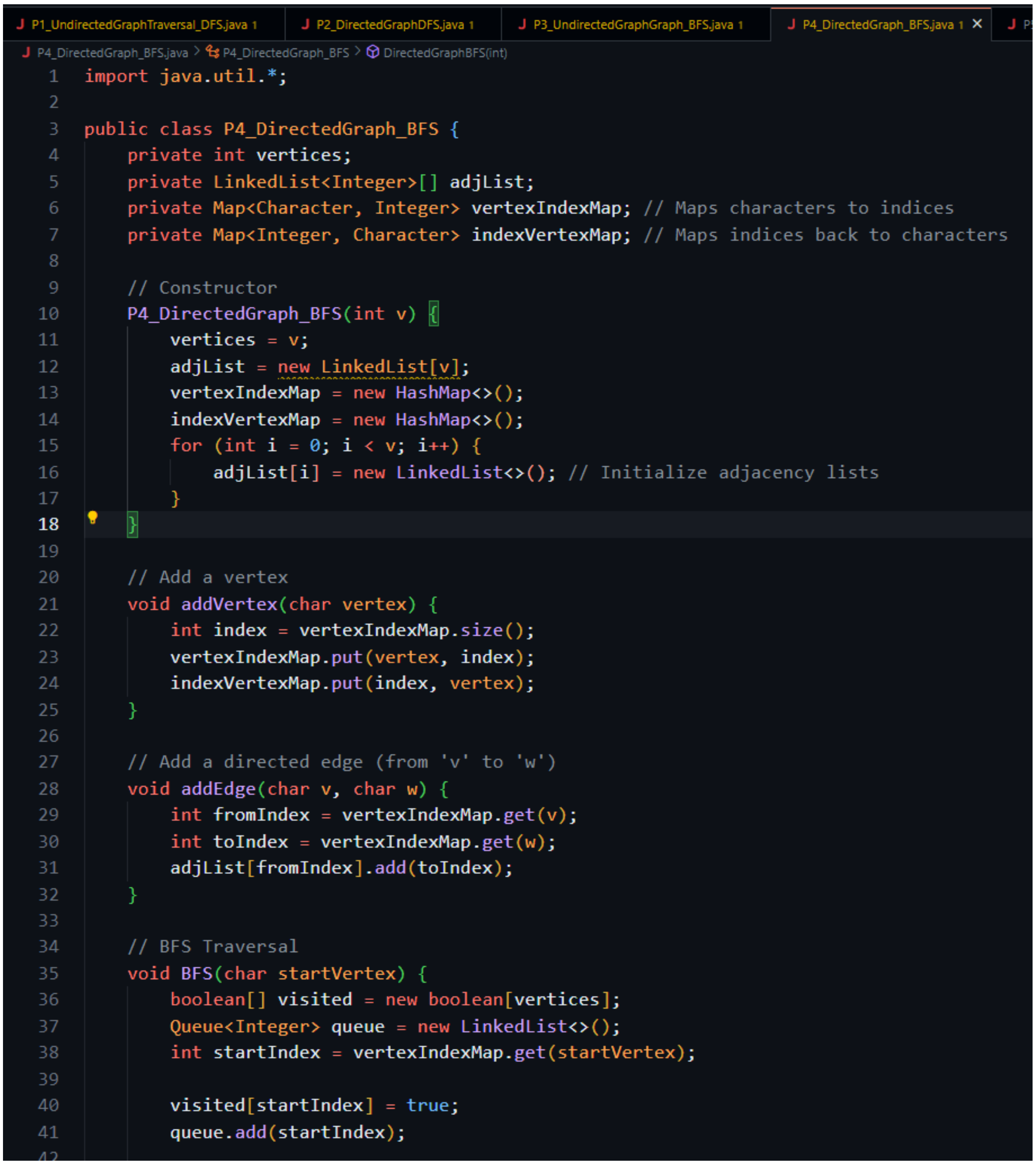
Output:



```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\31-03-202
P3_UndirectedGraphGraph_BFS
Note: P3_UndirectedGraphGraph_BFS.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Breadth-First Search (BFS) Traversal for an Undirected Graph:

BFS Traversal starting from vertex 0:
0 1 2 3 4 5
[Done] exited with code=0 in 0.739 seconds
```

Q4. Write Java program on Breadth First Search Traversal for directed graph

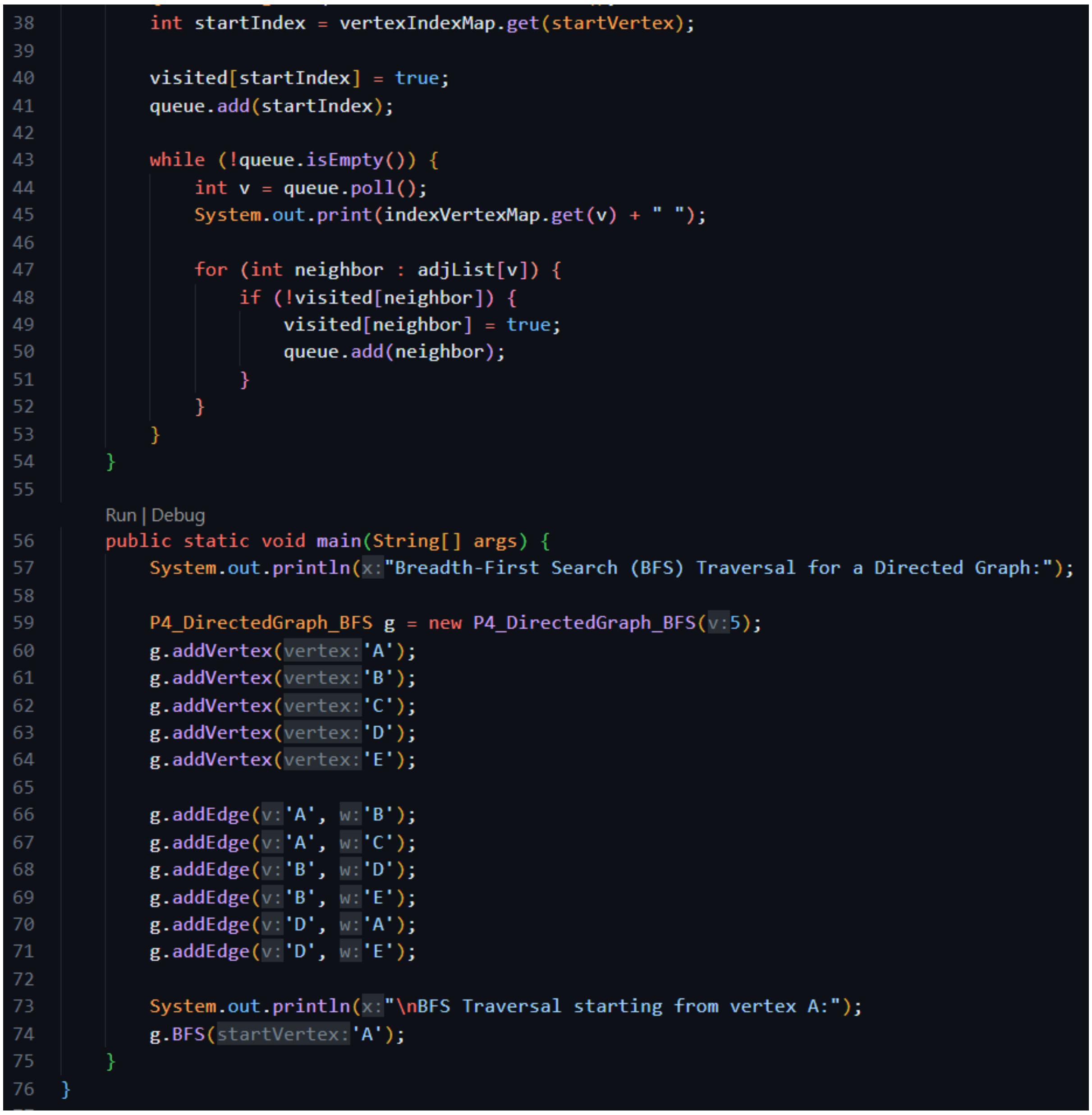
Solution:



P1_UndirectedGraphTraversal_DFS.java 1 | P2_DirectedGraphDFS.java 1 | P3_UndirectedGraphGraph_BFS.java 1 | P4_DirectedGraph_BFS.java 1

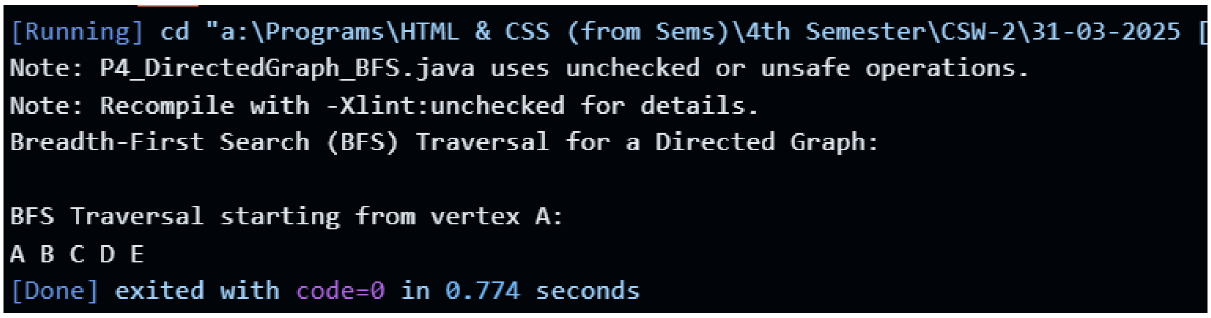
P4_DirectedGraph_BFS.java > P4_DirectedGraph_BFS > DirectedGraphBFS(int)

```java
import java.util.*;

public class P4_DirectedGraph_BFS {
    private int vertices;
    private LinkedList<Integer>[] adjList;
    private Map<Character, Integer> vertexIndexMap; // Maps characters to indices
    private Map<Integer, Character> indexVertexMap; // Maps indices back to characters

    // Constructor
    P4_DirectedGraph_BFS(int v) {
        vertices = v;
        adjList = new LinkedList[v];
        vertexIndexMap = new HashMap<>();
        indexVertexMap = new HashMap<>();
        for (int i = 0; i < v; i++) {
            adjList[i] = new LinkedList<>(); // Initialize adjacency lists
        }
    }

    // Add a vertex
    void addVertex(char vertex) {
        int index = vertexIndexMap.size();
        vertexIndexMap.put(vertex, index);
        indexVertexMap.put(index, vertex);
    }

    // Add a directed edge (from 'v' to 'w')
    void addEdge(char v, char w) {
        int fromIndex = vertexIndexMap.get(v);
        int toIndex = vertexIndexMap.get(w);
        adjList[fromIndex].add(toIndex);
    }

    // BFS Traversal
    void BFS(char startVertex) {
        boolean[] visited = new boolean[vertices];
        Queue<Integer> queue = new LinkedList<>();
        int startIndex = vertexIndexMap.get(startVertex);

        visited[startIndex] = true;
        queue.add(startIndex);
```



```java
        int startIndex = vertexIndexMap.get(startVertex);

        visited[startIndex] = true;
        queue.add(startIndex);

        while (!queue.isEmpty()) {
            int v = queue.poll();
            System.out.print(indexVertexMap.get(v) + " ");

            for (int neighbor : adjList[v]) {
                if (!visited[neighbor]) {
                    visited[neighbor] = true;
                    queue.add(neighbor);
                }
            }
        }
    }

    Run | Debug
    public static void main(String[] args) {
        System.out.println(x:"Breadth-First Search (BFS) Traversal for a Directed Graph:");

        P4_DirectedGraph_BFS g = new P4_DirectedGraph_BFS(v:5);
        g.addVertex(vertex:'A');
        g.addVertex(vertex:'B');
        g.addVertex(vertex:'C');
        g.addVertex(vertex:'D');
        g.addVertex(vertex:'E');

        g.addEdge(v:'A', w:'B');
        g.addEdge(v:'A', w:'C');
        g.addEdge(v:'B', w:'D');
        g.addEdge(v:'B', w:'E');
        g.addEdge(v:'D', w:'A');
        g.addEdge(v:'D', w:'E');

        System.out.println(x:"\nBFS Traversal starting from vertex A:");
        g.BFS(startVertex:'A');
    }
}
```

Output:

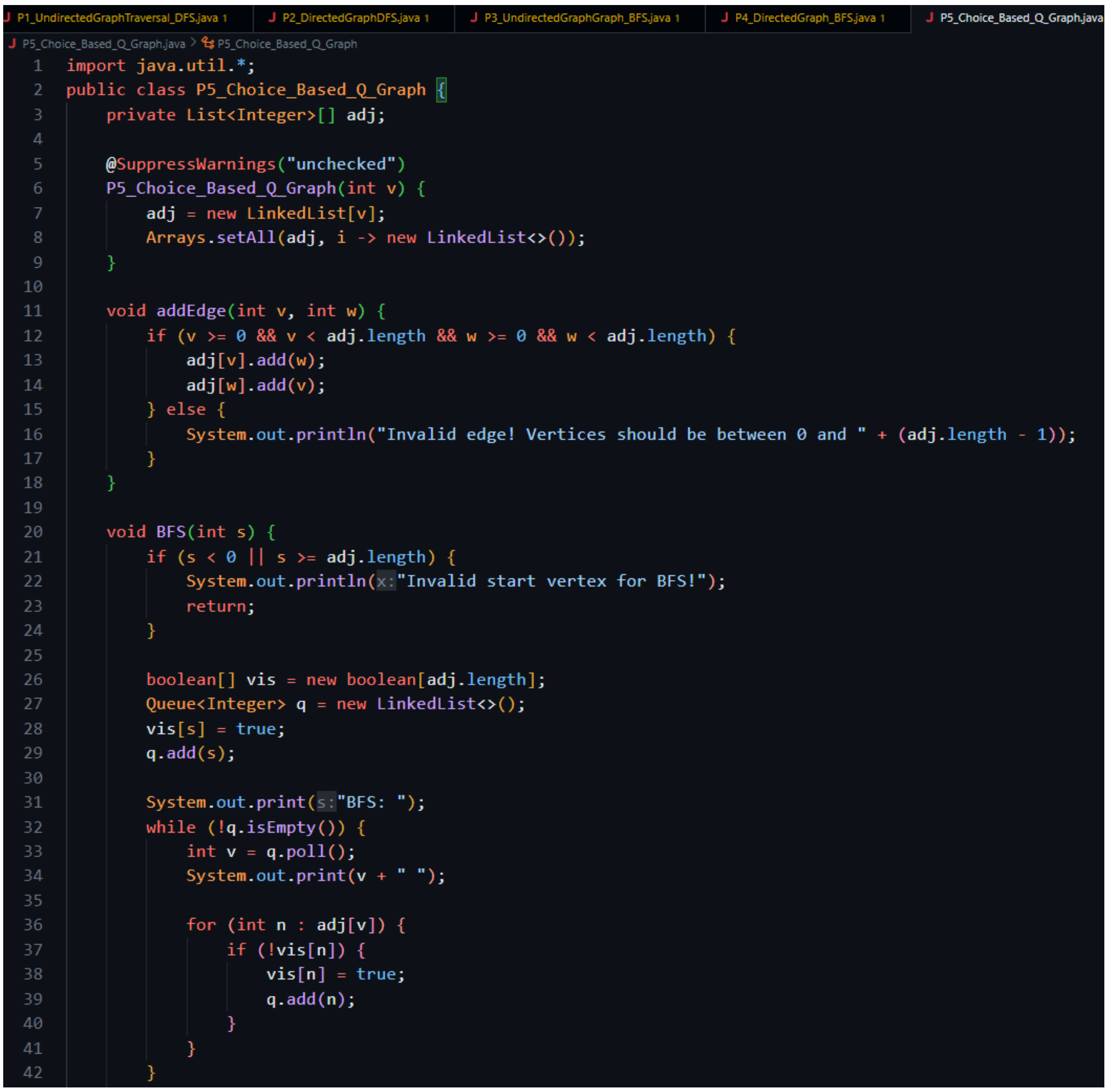


```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\31-03-2025 [
Note: P4_DirectedGraph_BFS.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Breadth-First Search (BFS) Traversal for a Directed Graph:

BFS Traversal starting from vertex A:
A B C D E
[Done] exited with code=0 in 0.774 seconds
```

Q5. Java Program on Graph Traversal using Switch case as: a) addEdge b) BFS c) DFS d) Exit default: invalid choice for the Undirected Graph. Use Scanner Class in the main function for choices in the run time and

adding the edges.

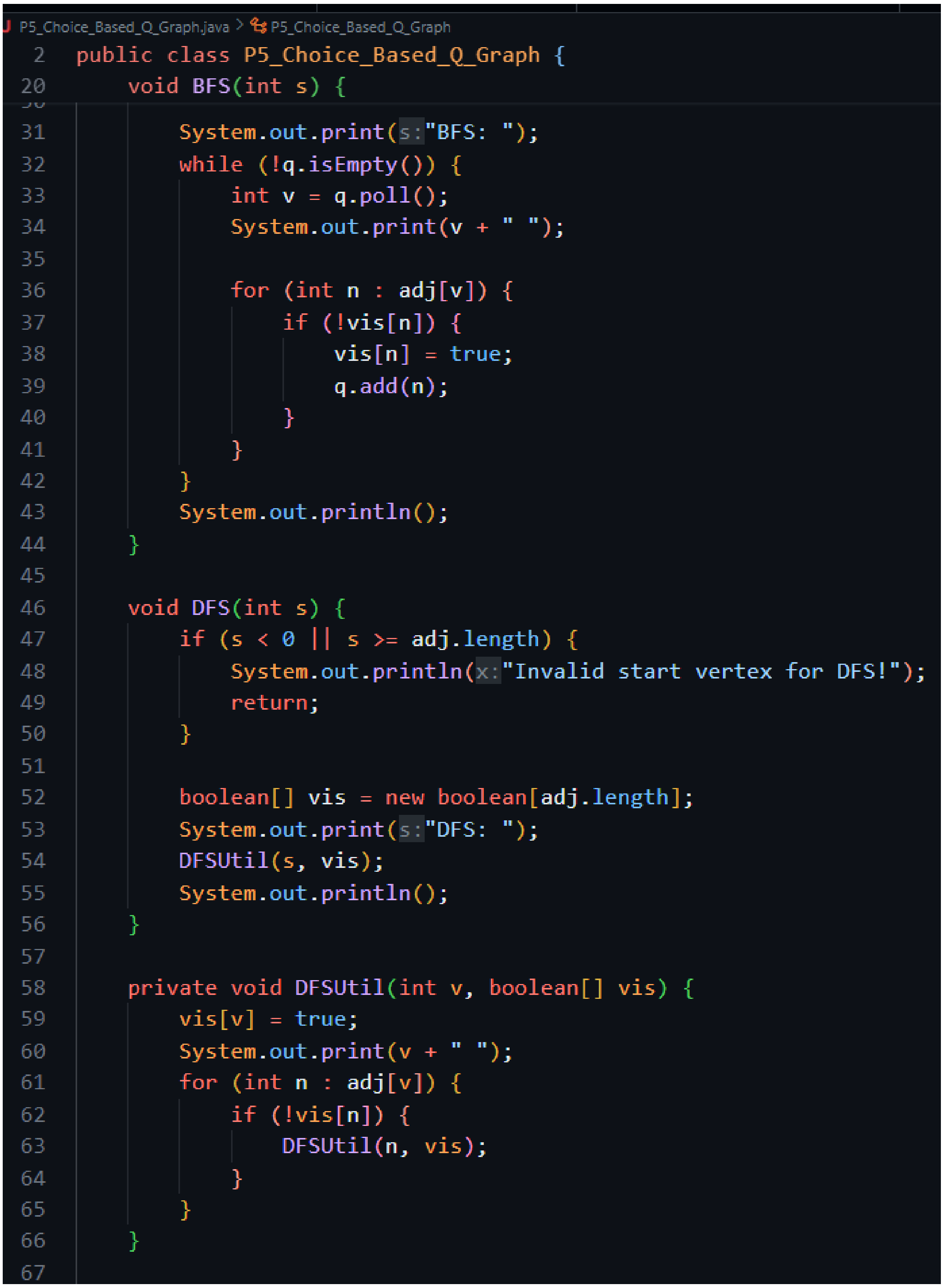
Solution:



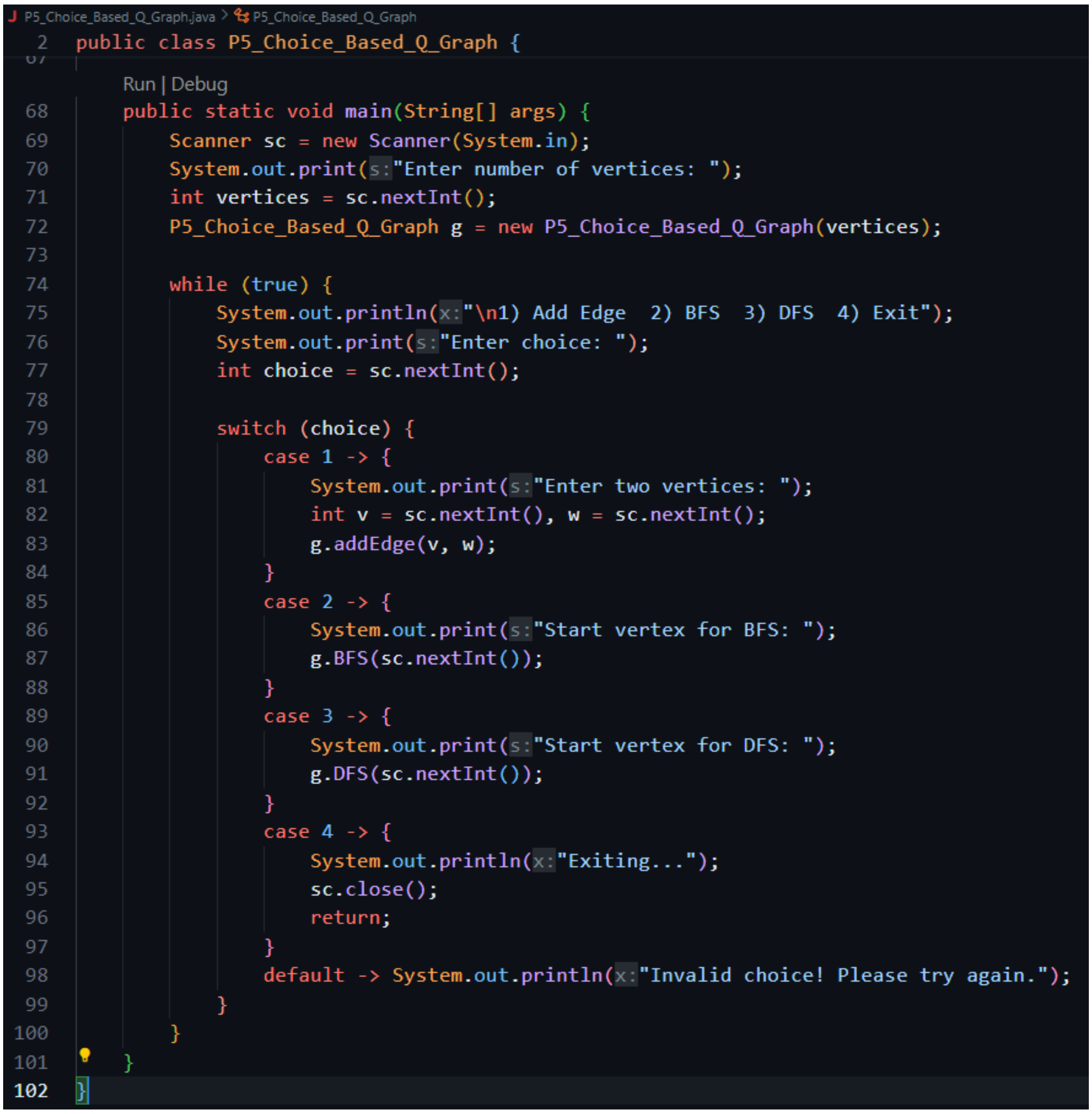
P1_UndirectedGraphTraversal_DFS.java 1 | P2_DirectedGraphDFS.java 1 | P3_UndirectedGraphGraph_BFS.java 1 | P4_DirectedGraph_BFS.java 1 | P5_Choice_Based_Q_Graph.java

P5_Choice_Based_Q_Graph.java > P5_Choice_Based_Q_Graph

```java
import java.util.*;
public class P5_Choice_Based_Q_Graph {
    private List<Integer>[] adj;

    @SuppressWarnings("unchecked")
    P5_Choice_Based_Q_Graph(int v) {
        adj = new LinkedList[v];
        Arrays.setAll(adj, i -> new LinkedList<>());
    }

    void addEdge(int v, int w) {
        if (v >= 0 && v < adj.length && w >= 0 && w < adj.length) {
            adj[v].add(w);
            adj[w].add(v);
        } else {
            System.out.println("Invalid edge! Vertices should be between 0 and " + (adj.length - 1));
        }
    }

    void BFS(int s) {
        if (s < 0 || s >= adj.length) {
            System.out.println(x:"Invalid start vertex for BFS!");
            return;
        }

        boolean[] vis = new boolean[adj.length];
        Queue<Integer> q = new LinkedList<>();
        vis[s] = true;
        q.add(s);

        System.out.print(s:"BFS: ");
        while (!q.isEmpty()) {
            int v = q.poll();
            System.out.print(v + " ");

            for (int n : adj[v]) {
                if (!vis[n]) {
                    vis[n] = true;
                    q.add(n);
                }
            }
        }
```

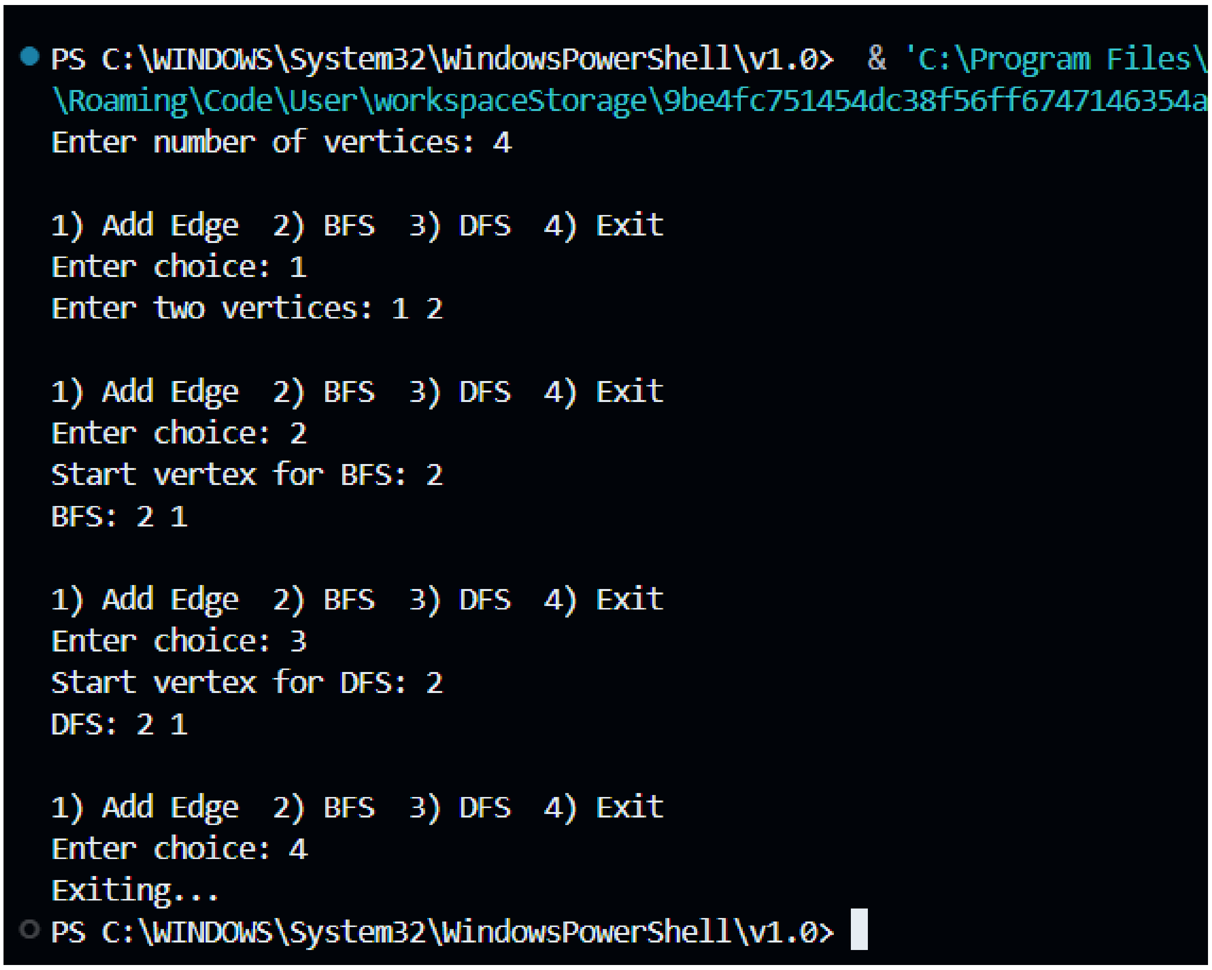

P5_Choice_Based_Q_Graph.java > P5_Choice_Based_Q_Graph

```java
public class P5_Choice_Based_Q_Graph {
    void BFS(int s) {
        System.out.print(s:"BFS: ");
        while (!q.isEmpty()) {
            int v = q.poll();
            System.out.print(v + " ");

            for (int n : adj[v]) {
                if (!vis[n]) {
                    vis[n] = true;
                    q.add(n);
                }
            }
        }
        System.out.println();
    }

    void DFS(int s) {
        if (s < 0 || s >= adj.length) {
            System.out.println(x:"Invalid start vertex for DFS!");
            return;
        }

        boolean[] vis = new boolean[adj.length];
        System.out.print(s:"DFS: ");
        DFSUtil(s, vis);
        System.out.println();
    }

    private void DFSUtil(int v, boolean[] vis) {
        vis[v] = true;
        System.out.print(v + " ");
        for (int n : adj[v]) {
            if (!vis[n]) {
                DFSUtil(n, vis);
            }
        }
    }

```


P5_Choice_Based_Q_Graph.java > P5_Choice_Based_Q_Graph

```java
public class P5_Choice_Based_Q_Graph {

    Run | Debug
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print(s:"Enter number of vertices: ");
        int vertices = sc.nextInt();
        P5_Choice_Based_Q_Graph g = new P5_Choice_Based_Q_Graph(vertices);

        while (true) {
            System.out.println(x:"\n1) Add Edge  2) BFS  3) DFS  4) Exit");
            System.out.print(s:"Enter choice: ");
            int choice = sc.nextInt();

            switch (choice) {
                case 1 -> {
                    System.out.print(s:"Enter two vertices: ");
                    int v = sc.nextInt(), w = sc.nextInt();
                    g.addEdge(v, w);
                }
                case 2 -> {
                    System.out.print(s:"Start vertex for BFS: ");
                    g.BFS(sc.nextInt());
                }
                case 3 -> {
                    System.out.print(s:"Start vertex for DFS: ");
                    g.DFS(sc.nextInt());
                }
                case 4 -> {
                    System.out.println(x:"Exiting...");
                    sc.close();
                    return;
                }
                default -> System.out.println(x:"Invalid choice! Please try again.");
            }
        }
    }
}
```

Output:



```
● PS C:\WINDOWS\System32\WindowsPowerShell\v1.0>  & 'C:\Program Files\
  \Roaming\Code\User\workspaceStorage\9be4fc751454dc38f56ff6747146354a
  Enter number of vertices: 4

  1) Add Edge  2) BFS  3) DFS  4) Exit
  Enter choice: 1
  Enter two vertices: 1 2

  1) Add Edge  2) BFS  3) DFS  4) Exit
  Enter choice: 2
  Start vertex for BFS: 2
  BFS: 2 1

  1) Add Edge  2) BFS  3) DFS  4) Exit
  Enter choice: 3
  Start vertex for DFS: 2
  DFS: 2 1

  1) Add Edge  2) BFS  3) DFS  4) Exit
  Enter choice: 4
  Exiting...
○ PS C:\WINDOWS\System32\WindowsPowerShell\v1.0>
```