JOBSHEET 16

Nama : Farid Aziz Wicaksono
Kelas : TI-1C
Absen : 14

A. Praktikum

1. Praktikum 1.

No	Graph.java
1	package jobsheet16;
2	import java.io.*;
3	import java.util.*;
4	
5	public class graph {
6	private int V;
7	private LinkedList<Integer> adj[];
8	
9	graph(int v) {
10	V = v;
11	adj = new LinkedList[v];
12	for (int i = 0; i < v; ++i) {
13	adj[i] = new LinkedList();
14	}
15	}
16	
17	void addEdge(int v, int w) {
18	adj[v].add(w);
19	}
20	
21	void BFS(int s) {
22	boolean visited[] = new boolean[V];
23	LinkedList<Integer> queue = new LinkedList<Integer>();
24	visited[s] = true;
25	queue.add(s);
26	while (queue.size() != 0) {
27	s = queue.poll();
28	System.out.print(s + " ");
29	Iterator<Integer> i = adj[s].listIterator();
30	while (i.hasNext()) {
31	int n = i.next();
32	if (!visited[n]) {
33	visited[n] = true;
34	queue.add(n);
35	}
36	}
37	}
38	}
39	}

```
40
41  class BFSGraph {
42      public static void main(String[] args) {
43          graph g = new graph(4);
44          g.addEdge(0, 1);
45          g.addEdge(0, 2);
46          g.addEdge(1, 2);
47          g.addEdge(2, 0);
48          g.addEdge(2, 3);
49          g.addEdge(2, 3);
50          System.out.println("Following is Breadth First Traversal"
51                  + "(starting from vertex 2)");
52          g.BFS(2);
53      }
54  }
```

Output :

```
run:
Following is Breadth First Traversal(starting from vertex 2)
2 0 3 1 BUILD SUCCESSFUL (total time: 0 seconds)
```

2. Praktikum 2

```
No
1   package jobsheet16;
2
3   import java.io.*;
4   import java.util.*;
5
6   public class dfsGraph {
7       private int V;
8       private LinkedList<Integer> adj[];
9
10      dfsGraph(int v) {
11          V = v;
12          adj = new LinkedList[v];
13          for (int i = 0; i < v; ++i) {
14              adj[i] = new LinkedList();
15          }
16      }
17
18      void addEdge(int v, int w) {
19          adj[v].add(w);
20      }
21
22      void DFSUtil(int v, boolean visited[]) {
23          visited[v] = true;
```

```
24          System.out.print(v + " ");
25          Iterator<Integer> i = adj[v].listIterator();
26          while (i.hasNext()) {
27              int n = i.next();
28              if (!visited[n]) {
29                  DFSUtil(n, visited);
30              }
31          }
32      }
33
34      void DFS(int v) {
35          boolean visited[] = new boolean[V];
36          DFSUtil(v, visited);
37      }
38  }
39
40  class main {
41      public static void main(String[] args) {
42          dfsGraph g = new dfsGraph(4);
43          g.addEdge(0, 1);
44          g.addEdge(0, 2);
45          g.addEdge(1, 2);
46          g.addEdge(2, 0);
47          g.addEdge(2, 3);
48          g.addEdge(3, 3);
49          System.out.println("Following is Depth First Traversal"
50                  + "(starting from vertex 2)");
51          g.DFS(2);
52      }
53  }
```

Output :

```
run:
Following is Depth First Traversal(starting from vertex 2)
2 0 1 3 BUILD SUCCESSFUL (total time: 0 seconds)
```

B. Pertanyaan

1. Apa perbedaan dari BFS dan DFS?
   Jawab :
   BFS mencari setiap solusi dalam grafik untuk memperluas nodusnya. Lubang DFS jauh di dalam nodus anak sampai tercapai tujuannya

2. Apa arti dari LinkedList<Integer>?
   Jawab :
   Menggunakan library LinkedList yang telah disediakan oleh Java untuk menampung data Integer

3. Mengapa kita perlu import java.util.* ?
   Jawab :
   Untuk mengimpor data *

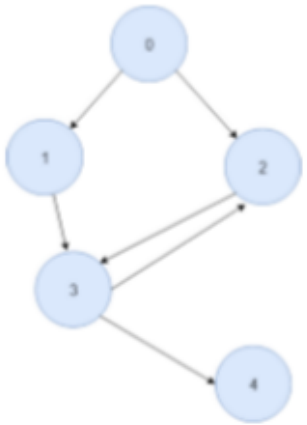
4. Pada program BFS, apa hasil BFS jika dimulai dari vertex 0?
   Jawab :

```
run:
Following is Breadth First Traversal(starting from vertex 2
0 1 2 3 BUILD SUCCESSFUL (total time: 0 seconds)
```

5. Pada program DFS, apa hasil DFS jika dimulai dari vertex 0?
   Jawab :

```
run:
Following is Depth First Traversal(starting from vertex 0)
0 1 2 3 BUILD SUCCESSFUL (total time: 0 seconds)
```

6. Perhatikan graph berikut ini:



7. Modifikasi program BFS pada class main agar tercipta graph dengan vertex dan edge seperti pada gambar diatas. Kemudian lakukan BFS dimulai dari vertex 0. Tulis output programnya.
   Jawab :

8. Modifikasi program DFS pada class main agar tercipta graph dengan vertex dan edge seperti pada gambar diatas. Kemudian lakukan DFS dimulai dari vertex 0. Tulis output programnya.

C. Tugas

1. Modifikasi program BFS agar dapat menerima input jumlah vertex pada graph dan edge-edge nya. Dan juga menerima input start vertex-nya.

2. Modifikasi program DFS agar dapat menerima input jumlah vertex pada graph dan edge-edge nya. Dan juga menerima input start vertex-nya.

3. Gabungkan program BFS dan DFS, sediakan menu:
   - Buat graph
   - Lakukan BFS
   - Lakukan DFS

4. (Opsional) Implementasikan BFS pada program untuk mencari jalan dari kota-kota berikut ini: