W3D5_Answers

1,

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0								7			1			1	1	
1			1							1				1		
2											1		1	1	1	
3							1		1	1		1				
4								1								
5							1	1		1	1					
6																1
7															1	
8																1
9												1				
10															1	
11																
12																
13																
14																
15																

2, 3,5,6,8,0,1,2,10,4,9,11,14,15

3,

```
// A Java program to print topological sorting of a DAG
import java.io.*;
import java.util.*;

// This class represents a directed graph using adjacency
// list representation
class Graph
{
    private int V;   // No. of vertices
    private LinkedList<Integer> adj[]; // Adjacency List

    //Constructor
    Graph(int v)
    {
        V = v;
        adj = new LinkedList[v];
        for (int i=0; i<v; ++i)
            adj[i] = new LinkedList();
    }

    // Function to add an edge into the graph
    void addEdge(int v,int w) { adj[v].add(w); }

    // A recursive function used by topologicalSort
    void topologicalSortUtil(int v, boolean visited[],
                         Stack stack)
{
    // Mark the current node as visited.
    visited[v] = true;
    Integer i;

    // Recur for all the vertices adjacent to this
    // vertex
    Iterator<Integer> it = adj[v].iterator();
    while (it.hasNext())
    {
        i = it.next();
        if (!visited[i])
            topologicalSortUtil(i, visited, stack);
    }

    // Push current vertex to stack which stores result
    stack.push(new Integer(v));
}

// The function to do Topological Sort. It uses
// recursive topologicalSortUtil()
void topologicalSort()
{
    Stack stack = new Stack();

    // Mark all the vertices as not visited
    boolean visited[] = new boolean[V];
    for (int i = 0; i < V; i++)
        visited[i] = false;

    // Call the recursive helper function to store
    // Topological Sort starting from all vertices
    // one by one
    for (int i = 0; i < V; i++)
        if (visited[i] == false)
            topologicalSortUtil(i, visited, stack);

    // Print contents of stack
    while (stack.empty()==false)
        System.out.print(stack.pop() + " ");
}

// Driver method
public static void main(String args[])
{
    // Create a graph given in the above diagram
    Graph g = new Graph(6);
    g.addEdge(5, 2);
    g.addEdge(5, 0);
    g.addEdge(4, 0);
    g.addEdge(4, 1);
    g.addEdge(2, 3);
    g.addEdge(3, 1);
```

```
        System.out.println("Following is a Topological " +
                           "sort of the given graph");
        g.topologicalSort();
    }
}
// This code is contributed by Aakash Hasija
```