FLAT ASSIGNMENT

#implementation of topological sorting:

program to print topological sorting

from collections import defaultdict

Class to represent a graph

#Algorithm

class Graph:

    def __init__(self, vertices):

        self.graph = defaultdict(list) # dictionary containing adjacency List

        self.V = vertices # No. of vertices

def addEdge(self, u, v):

    self.graph[u].append(v)

A recursive function used by topologicalSort

def topologicalSortUtil(self, v, visited, stack):

    # Mark the current node as visited.

    visited[v] = True

    # Recur for all the vertices adjacent to this vertex

    for i in self.graph[v]:

        if visited[i] == False:

self.topologicalSortUtil(i, visited, stack)

Push current vertex to stack which stores result

stack.append(v)

The function to do Topological Sort. It uses recursive

#topologicalSortUtil()

#Program:

def topologicalSort(self):

Mark all the vertices as not visited

visited = [False]*self.V

stack = []

Call the recursive helper function to store Topological

Sort starting from all vertices one by one

for i in range(self.V):

if visited[i] == False:

self.topologicalSortUtil(i, visited, stack)

Print contents of the stack

print(stack[::-1]) # return list in reverse order

```
# Driver Code
if __name__ == '__main__':
        g = 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)
        print("Following is a Topological Sort of the given graph")


        # Function Call
        g.topologicalSort()
```

output:

```
Following is a Topological Sort of the given graph
[5, 4, 2, 3, 1, 0]
```

Submitted by:

J.SHARATH REDDY

21071A6720