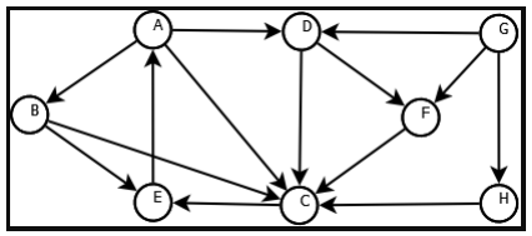
**TASK 1:**

**Apply DFS search on the graph given below.**

****

**CODE:**

def dfs(graph,start,path=[]):

    stack=[start]

    while stack:

        v=stack.pop(0)

        if not v in path:

            path=path+[v]

            stack=graph[v]+stack

    return path

graph={ 'A':['B','C','D'],

        'B':['C','E'],

        'C':['E'],

        'D':['C','F'],

        'E':['A'],

        'F':['C'],

        'G':['D','F','H'],

        'H':['C']}

print('Depth First Search starting from A',dfs(graph,'A'))

print('\nDFS starting from B',dfs(graph,'B'))

print('\nDFS First Search starting from C',dfs(graph,'C'))

print('\nDFS First Search starting from D',dfs(graph,'D'))

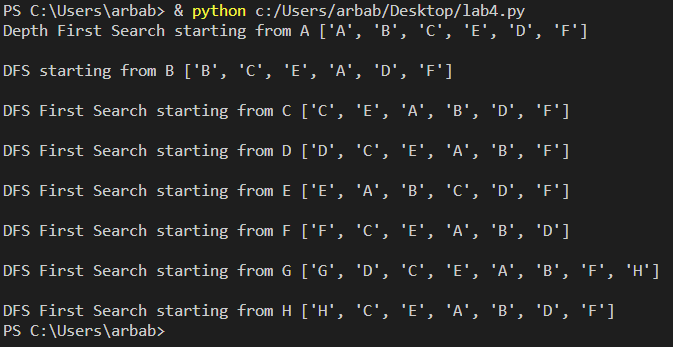
print('\nDFS First Search starting from E',dfs(graph,'E'))

print('\nDFS First Search starting from F',dfs(graph,'F'))

print('\nDFS First Search starting from G',dfs(graph,'G'))

print('\nDFS First Search starting from H',dfs(graph,'H'))

**OUTPUT:**

****

**TASK 2:**

**Implement DFS search algorithm while using recursion**

**CODE:**

def dfs(graph,start,path=[]):

    path=path+[start]

    for node in graph[start]:

        if not node in path:

            path=dfs(graph,node,path)

    return path

graph={ 'A':['C','D','G'],

        'B':['C','D','E'],

        'C':['A','B','F'],

        'D':['A','B','G'],

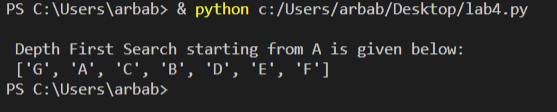
        'E':['B'],

        'F':['C'],

        'G':['A','D']}

print('\n Depth First Search starting from A is given below:\n',dfs(graph,'G'))

**OUTPUT:**

****