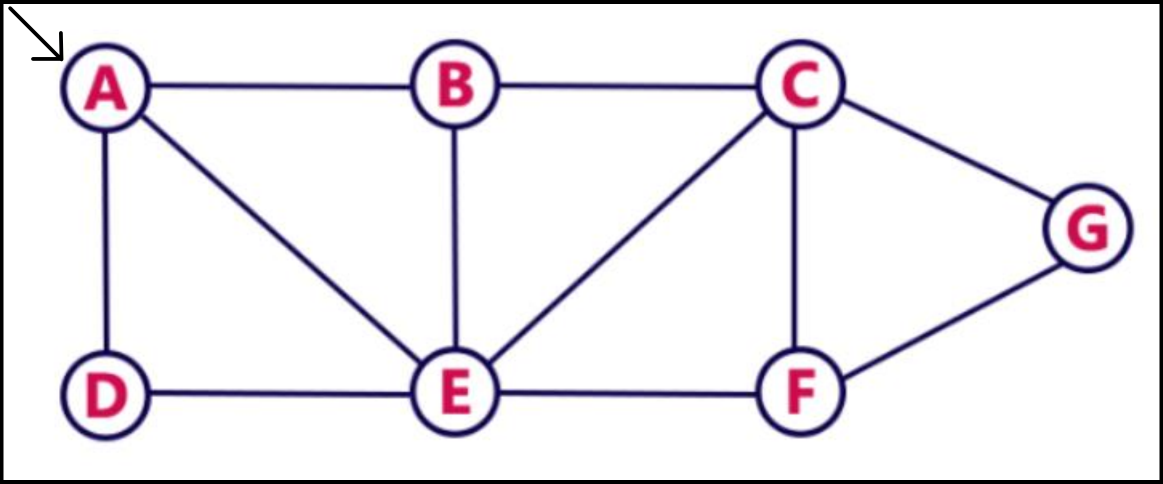
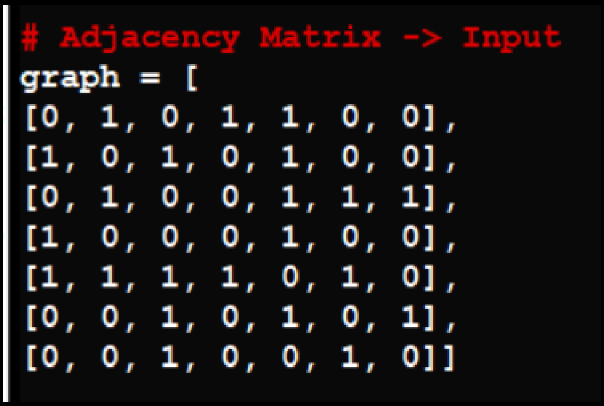
Name: Girish S **Roll No.: AM.EN.U4AIE22044**

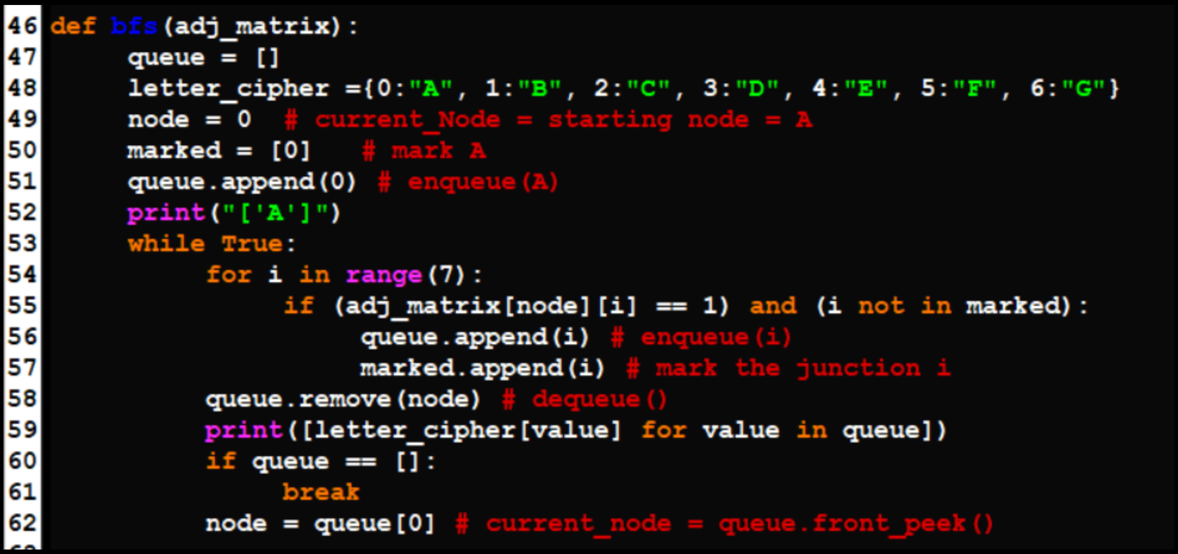
DATA STRUCTURES AND ALGORITHMS -2



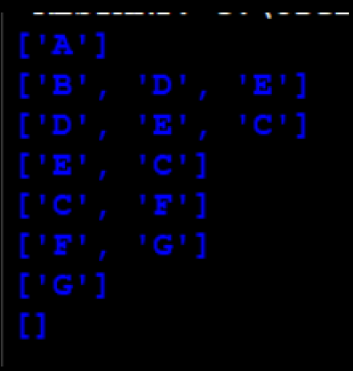


```
# Adjacency Matrix -> Input
graph = [
[0, 1, 0, 1, 1, 0, 0],
[1, 0, 1, 0, 1, 0, 0],
[0, 1, 0, 0, 1, 1, 1],
[1, 0, 0, 0, 1, 0, 0],
[1, 1, 1, 1, 0, 1, 0],
[0, 0, 1, 0, 1, 0, 1],
[0, 0, 1, 0, 0, 1, 0]]
```

1. Perform Breadth First Search Algorithm on the given Graph using Adjacency matrix or Adjacency list

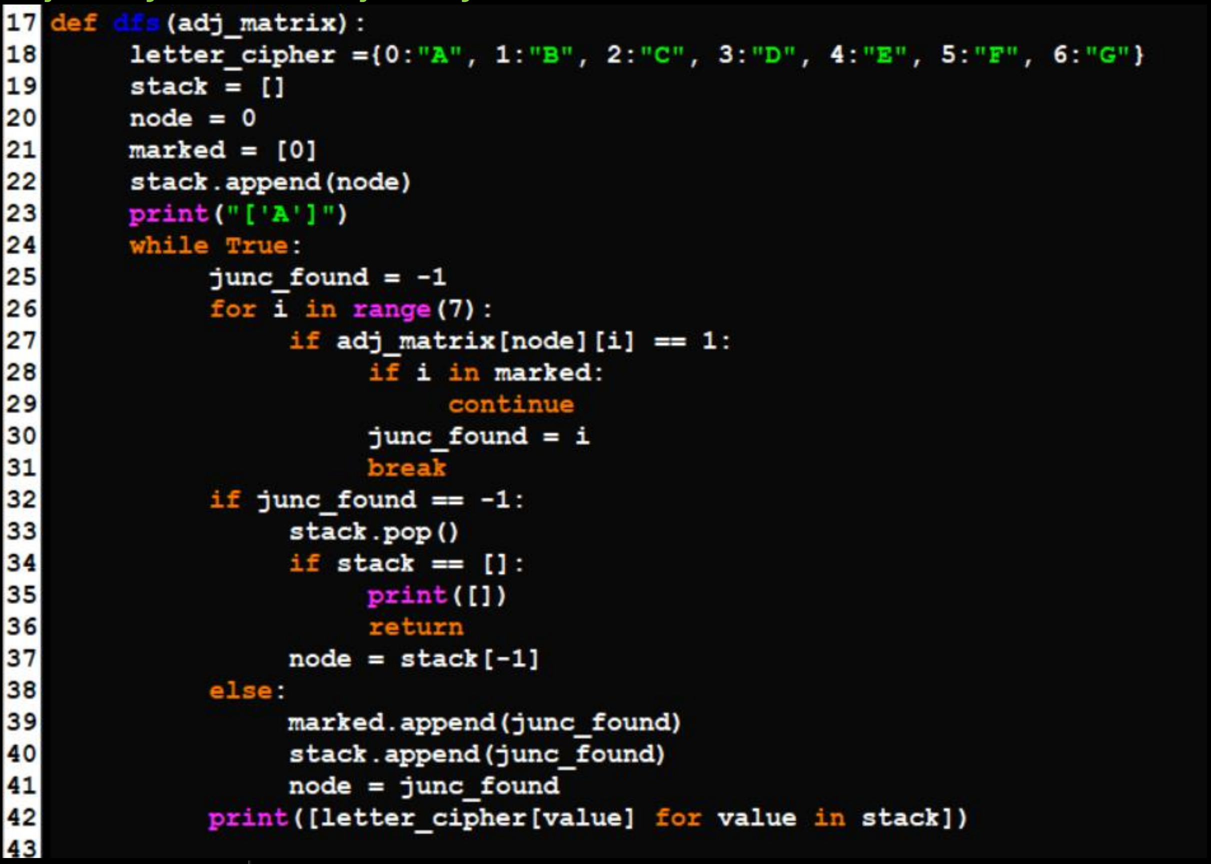


```
def bfs(adj_matrix):
    queue = []
    letter_cipher ={0:"A", 1:"B", 2:"C", 3:"D", 4:"E", 5:"F", 6:"G"}
    node = 0  # current_Node = starting node = A
    marked = [0]   # mark A
    queue.append(0) # enqueue(A)
    print("['A']")
    while True:
        for i in range(7):
            if (adj_matrix[node][i] == 1) and (i not in marked):
                queue.append(i) # enqueue(i)
                marked.append(i) # mark the junction i
        queue.remove(node) # dequeue()
        print([letter_cipher[value] for value in queue])
        if queue == []:
            break
        node = queue[0] # current_node = queue.front_peek()
```

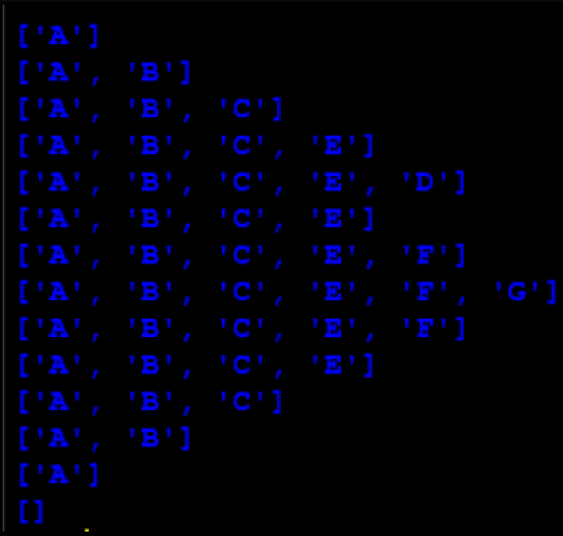


```
['A']
['B', 'D', 'E']
['D', 'E', 'C']
['E', 'C']
['C', 'F']
['F', 'G']
['G']
[]
```

2. **Perform Depth First Search Algorithm on the given Graph using Adjacency matrix or Adjacency list**



```python
def dfs(adj_matrix):
    letter_cipher ={0:"A", 1:"B", 2:"C", 3:"D", 4:"E", 5:"F", 6:"G"}
    stack = []
    node = 0
    marked = [0]
    stack.append(node)
    print("['A']")
    while True:
        junc_found = -1
        for i in range(7):
            if adj_matrix[node][i] == 1:
                if i in marked:
                    continue
                junc_found = i
                break
        if junc_found == -1:
            stack.pop()
            if stack == []:
                print([])
                return
            node = stack[-1]
        else:
            marked.append(junc_found)
            stack.append(junc_found)
            node = junc_found
        print([letter_cipher[value] for value in stack])
```



```
['A']
['A', 'B']
['A', 'B', 'C']
['A', 'B', 'C', 'E']
['A', 'B', 'C', 'E', 'D']
['A', 'B', 'C', 'E']
['A', 'B', 'C', 'E', 'F']
['A', 'B', 'C', 'E', 'F', 'G']
['A', 'B', 'C', 'E', 'F']
['A', 'B', 'C', 'E']
['A', 'B', 'C']
['A', 'B']
['A']
[]
```