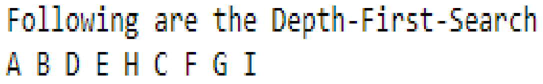
Group A: Assignment No. 1

Title :- Implement depth first search algorithm and breadth first search algorithm, Use an undirected graph and develop recursive algorithm for searching all the vertices of a graph or tree data structure.

DFS Code:-

```
graph = {
'A':['B','C'],  'B':['D','E'], 'C':['F','G'], 'D':[], 'E':['H'], 'F':[], 'G':['I'], 'H':[], 'I':[]}
visited =set()
def dfs1(visited,graph,root):
    print(root,end=" ")
    visited.add(root)
    for neighbour in graph[root]:
        dfs1(visited,graph,neighbour)
print("Following are the Depth-First-Search")
dfs1(visited,graph,'A')
```

Output:-

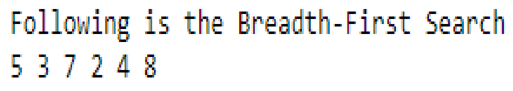


```
Following are the Depth-First-Search
A B D E H C F G I
```

BFS Code:-

```
#Implementation Of BFS

graph = {

  '5' : ['3','7'], '3': ['2', '4'], '7': ['8'], '2': [], '4': ['8'], '8': []}

visited = [] # List for visited nodes.

queue = []     #Initialize a queue

def bfs(visited, graph, node): #function for BFS

  visited.append(node)

  queue.append(node)

  while queue:          # Creating loop to visit each node

    m = queue.pop(0)

    print (m, end = " ")

    for neighbour in graph[m]:

      if neighbour not in visited:

        visited.append(neighbour)

        queue.append(neighbour)

# Driver Code

print("Following is the Breadth-First Search")

bfs(visited, graph, '5')    # function calling
```

Output:-



```
Following is the Breadth-First Search
5 3 7 2 4 8
```