**TASK:1**

Implementation of Graph search algorithms (**Breadth first search and Depth First Search**) using following constraints.

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*

The output of the above code will be as follow:

 Following is the Breadth-First Search

 5 3 7 2 4 8

graph = {

'5' : ['3','7'],

'3' : ['2', '4'],

'7' : ['8'],

'2' : [],

'4' : ['8'],

'8' : []

}

visited = set() # Set to keep track of visited nodes of graph.

**def** **dfs**(visited, graph, node): #function for dfs

**if** node **not** **in** visited:

**print** (node)

visited.add(node)

**for** neighbour **in** graph[node]:

dfs(visited, graph, neighbour)

# Driver Code

**print**("Following is the Depth-First Search")

dfs(visited, graph, '5')

Output

The output of the above code is as follow:

Following is the Depth-First Search

5 3 2 4 8 7