**Task 1**

def iterative\_preorder(root):

if root is None:

return

stack = [root]

while stack:

node = stack.pop()

print(node.value, end=" ")

if node.right:

stack.append(node.right)

if node.left:

stack.append(node.left)

print("\nIterative Preorder Traversal:")

iterative\_preorder(root)

def bfs(graph, start):

visited = set()

queue = [start]

while queue:

node = queue.pop(0)

if node not in visited:

print(node, end=" ")

visited.add(node)

for neighbor in graph[node]:

if neighbor not in visited:

queue.append(neighbor)

graph = {

1: [2, 3],

2: [1, 4, 5],

3: [1, 6],

4: [2],

5: [2],

6: [3]

}

print("BFS Traversal:")

**Task. 2**

from collections import deque

class Node:

def \_\_init\_\_(self, value):

self.value = value

self.neighbors = []

self.visited = False

def add\_neighbor(self, neighbor):

self.neighbors.append(neighbor)

def bfs(start\_node):

queue = deque([start\_node])

start\_node.visited = True

while queue:

current\_node = queue.popleft()

print(current\_node.value, end=" ")

for neighbor in current\_node.neighbors:

if not neighbor.visited:

neighbor.visited = True

queue.append(neighbor)

if \_\_name\_\_ == "\_\_main\_\_":

nodeA = Node('A')

nodeB = Node('B')

nodeC = Node('C')

nodeD = Node('D')

nodeE = Node('E')

nodeA.add\_neighbor(nodeB)

nodeA.add\_neighbor(nodeC)

nodeB.add\_neighbor(nodeA)

nodeB.add\_neighbor(nodeD)

nodeC.add\_neighbor(nodeA)

nodeD.add\_neighbor(nodeB)

nodeD.add\_neighbor(nodeE)

nodeE.add\_neighbor(nodeD)

print("BFS Traversal starting from Node A:")

bfs(nodeA)