# TASK 6

# BFS (Breadth-First Search) Code Explanation

## Code

tree = {  
 'A': ['B', 'C'],  
 'B': ['D', 'E'],  
 'C': ['F'],  
 'D': [],  
 'E': [],  
 'F': ['G'],  
 'G': []  
}  
  
def bfs(start, goal):  
 visited = []  
 queue = [start]  
  
 while queue:  
 current = queue.pop(0)  
 if current not in visited:  
 visited.append(current)  
 if current == goal:  
 return visited  
 queue.extend(tree[current])  
 return visited  
  
print(bfs('A', 'F'))

## Explanation:

This program demonstrates the Breadth-First Search (BFS) algorithm using a queue. BFS explores nodes level by level, starting from the given starting node.  
  
1. A tree structure is represented as a dictionary where each node is a key and its value is a list of children.  
2. The bfs function takes two parameters: the starting node and the goal node.  
3. A visited list keeps track of the nodes that have already been visited.  
4. A queue is used to explore nodes in a FIFO (First-In-First-Out) manner.  
5. The algorithm repeatedly dequeues the first element, checks if it is visited, and if not, marks it as visited.  
6. If the current node is the goal, the function returns the list of visited nodes up to that point.  
7. Otherwise, the neighbors of the current node are added to the queue for further exploration.  
8. Finally, if the goal is not found, the function returns all visited nodes.

## Functionality:

In this example, the BFS traversal starts at node 'A' and searches for node 'F'. The order of exploration will be:  
  
Visited nodes: ['A', 'B', 'C', 'D', 'E', 'F']  
  
Since 'F' is found during traversal, the function returns the visited list up to 'F'.  
  
Output:  
['A', 'B', 'C', 'D', 'E', 'F']