Task 6

Explanation:

First Search (BFS) in two ways

1. Recursive BFS (without a queue)
2. BFS using a queue (iterative approach)

1. Recursive BFS (without a queue)

This approach uses recursion instead of a queue.

How it work in easy words.

The function bfs_recursive(graph, node, visited=None, result=None):

Takes a graph (dictionary), a starting node, and two optional parameters (visited and result).

Initializes visited (to keep track of visited nodes) and result (to store traversal order).

If the node is not visited, it adds it to result and marks it as visited.

It then recursively calls itself for all neighboring nodes.

For this graph:

```
Graph = {
  'A': ['B', 'C'],
  'B': ['D', 'E'],
  'C': ['F'],
  'D': [],
  'E': [],
  'F': []
}
```

Calling bfs_recursive(graph, ‘A’) would return:

[‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’]

This follows BFS order: A → B → C → D → E → F

2. BFS using a Queue

This approach follows the standard BFS algorithm using a queue (FIFO – First In, First

Uses a deque (double-ended queue) from Python’s collections module.

The function bfs_queue(root):

Starts with the root node in the queue.

Uses a loop to process each node, printing its value.

Adds unvisited child nodes to the queue.

Continues until all nodes are visited.