**Name AQSA GILLANI  
Roll no 064  
Class BSDS-3A**

**Question no 1:**

**Explanation of BFS Code**

This Python code performs a **Breadth First Search (BFS)** traversal on a graph.  
Below is a step-by-step explanation:

**1. Function Definition**

* The function bfs(graph, start, visited=None) is defined to perform BFS.
* Parameters:
  + **graph** → A dictionary representing the graph.
  + **start** → The starting node for traversal.
  + **visited** → A set to keep track of visited nodes.

**2. Initialize Visited Set**

* If visited is None, an empty set is created.
* This ensures we don’t visit the same node more than once.

**3. Initialize Current Level**

* current\_level stores nodes at the current step of BFS.
* It starts with the given start node.
* The start node is marked as visited.

**4. While Loop**

* The loop runs while there are nodes in current\_level.
* A new list next\_level is created to store neighbors for the next step.

**5. Visit Nodes in Current Level**

* Each node in current\_level is printed.
* This shows the BFS traversal order.

**6. Explore Neighbors**

* For each neighbor of the current node:
  + If the neighbor has **not** been visited:
    - Add it to the visited set.
    - Append it to next\_level.

**7. Move to Next Level**

* After finishing the current level, current\_level is updated to next\_level.
* The process repeats until no nodes remain.

**Output:**

****

**Question no 2:**

Explanation of BFS Code (Using Queue)

This Python code performs a Breadth First Search (BFS) traversal on a graph.  
Below is a step-by-step explanation:

1. Function Definition

* The function bfs(graph, start) is defined to perform BFS.
* Parameters:
  + graph → A dictionary representing the graph.
  + start → The starting node for traversal.

2. Initialize Data Structures

* visited → A set that keeps track of visited nodes.
* queue → A list used as a queue (FIFO). It starts with the start node.

3. While Loop

* The loop runs while there are nodes in the queue.
* This ensures BFS continues until all reachable nodes are visited.

4. Dequeue a Node

* The first element is removed from the queue (queue.pop(0)).
* This ensures BFS works in FIFO order (first-in, first-out).

5. Visit Node

* If the node has not been visited:
  + Print the node (showing traversal order).
  + Add the node to the visited set.

6. Explore Neighbors

* For each neighbor of the current node:
  + If the neighbor has not been visited:
    - Add it to the queue.

7. Traversal Ends

* Once the queue becomes empty, BFS stops.
* The output shows the order in which nodes were visited

**Output:**

****