Theory Questions and Short Answers on Graph Traversal with OpenMP

1. What is the purpose of the Graph class in the code?

It represents a graph using an adjacency list and provides functions for adding edges, BFS, and DFS traversal.

2. What data structure is used to store the graph?

An adjacency list implemented using a vector of vector<int>.

3. How is an undirected edge added between two vertices?

By adding each vertex to the other's adjacency list using push_back.

4. What traversal algorithm uses a queue in this code?

Breadth-First Search (BFS).

5. What traversal algorithm uses recursion in this code?

Depth-First Search (DFS).

6. What is OpenMP?

A parallel programming API used to write multi-threaded programs in C/C++ and Fortran.

7. Which directive is used to parallelize a loop in OpenMP?

#pragma omp parallel for.

8. Why is #pragma omp critical used in the BFS function?

To prevent multiple threads from modifying the queue and visited array at the same time.

9. Why might the output order of BFS/DFS change with OpenMP?

Because OpenMP runs certain parts of the code in parallel, and thread scheduling is not deterministic.

10. What are potential issues when using OpenMP in graph traversal?

Race conditions and incorrect results if shared data is not handled safely.

11. How is parallelism introduced in the DFS function?

By using #pragma omp parallel and #pragma omp single for the initial call, and #pragma omp parallel for inside the DFS utility function.

12. What is the role of the visited array?

To keep track of which nodes have already been visited to avoid reprocessing.

13. Can OpenMP make every program faster? Why or why not?

No, because parallelism overhead can outweigh the benefits for small or I/O-bound tasks.

14. Is the DFS implementation thread-safe in this code?

No, because multiple threads can write to visited simultaneously without synchronization.

15. What is the time complexity of BFS and DFS in this code?

O(V + E), where V is the number of vertices and E is the number of edges.