Tree Traversals

- **treeDFS(TreeNode* root) → *Pre-order DFS (Recursive)***
  - Visit root → left → right.
  - Base case: if root == NULL, return.
  - Print the current node's value.
  - Recursively call treeDFS() on left and right children

Example output for tree:

```
  1
 / \
2   3
/ \
4   5
```

treeBFS(TreeNode* root) → *Level-order BFS (Iterative using Queue)*

- Uses a queue to process nodes level by level.
- Push root into the queue.
- While queue is not empty:
  - Pop the front node and print it.
  - Push its left and right children (if exist).

Graph Traversals

- **graphDFS(int start) → *DFS using Stack (Iterative)***
  - Initialize a visited[] array and a stack.
  - Push the start node into the stack.
  - While the stack is not empty:
    - Pop the top node.
    - If not visited, mark as visited and print it.
    - Push its neighbors **in reverse order** to match recursive DFS order.

Graph Used:

0

| \

1 2

| \

3   4

graphBFS(int start) → *BFS using Queue (Iterative)*

- Initialize visited[] and a queue.
- Push the start node and mark it visited.
- While the queue is not empty:
    - Pop the front node and print it.
    - For all unvisited neighbors, mark and enqueue them.

Key Concepts Used

- Tree vs Graph structures.
- **Stack** for DFS (LIFO), **Queue** for BFS (FIFO).
- Recursive and Iterative traversal.
- Use of adjacency list (vector<vector<int>>) for graph representation.