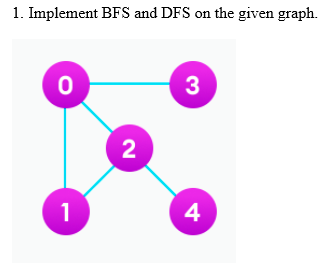
*LAB # 12*

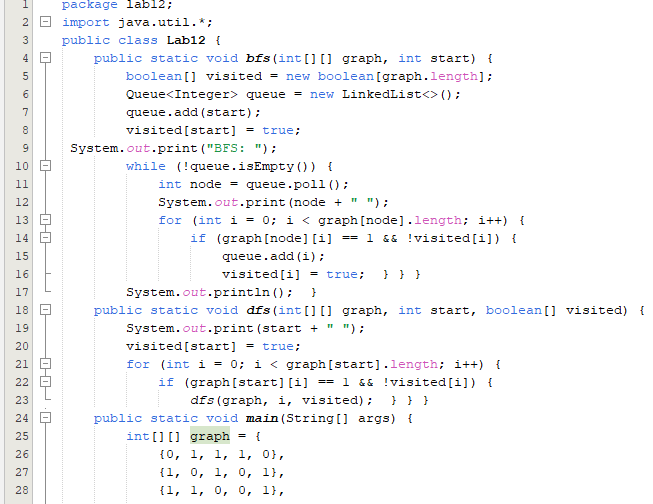
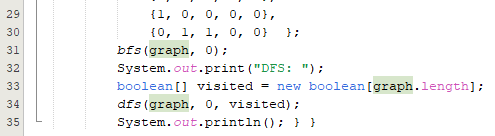
GRAPH AND GRAPH TRAVERSAL

# *OBJECTIVE:*

*implement graph using BFS and DFS*

*LAB task*

******

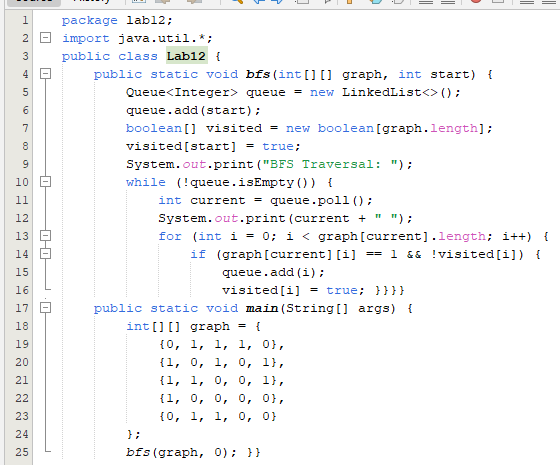
************

**

1. *Follow the below instructions to implement BFS traversal.*
2. *Declare a queue and insert the starting vertex.*
3. *Initialize a visited array and mark the starting vertex as visited.*

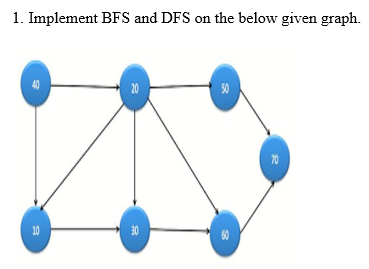
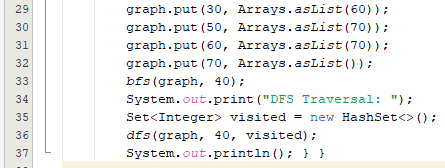
*Follow the below process till the queue becomes empty:*

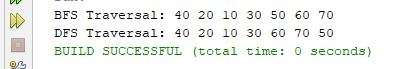
1. *Remove the first vertex of the queue.*
2. *Mark that vertex as visited.*
3. *Insert all the unvisited neighbors of the vertex into the queue.*

**

**

***HOME TASKS***

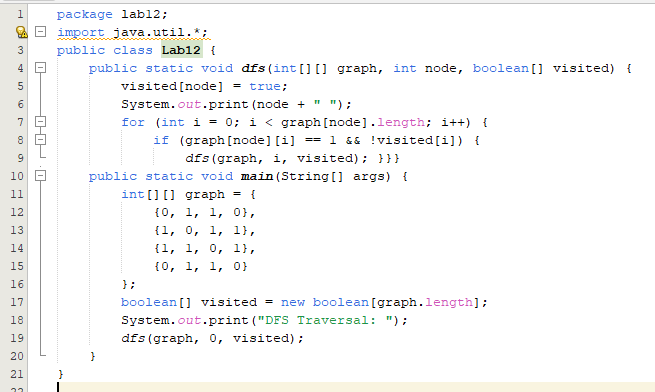
** ******

****

1. *Write a JAVA program that creates a recursive function that takes the index of the node and a visited array. Mark the current node as visited and print the node and traverse all the adjacent and unmarked nodes and call the recursive function with the index of the adjacent node.*

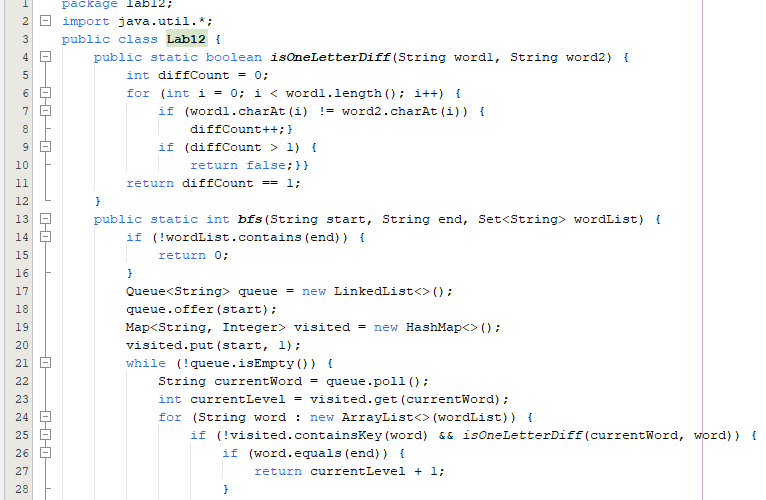
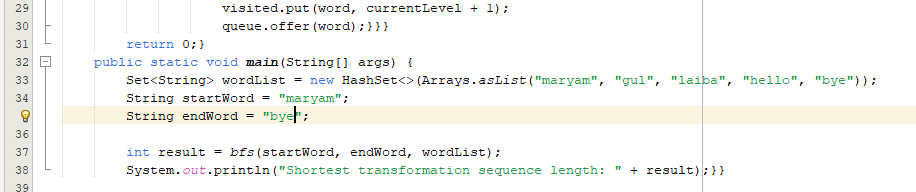
*.*

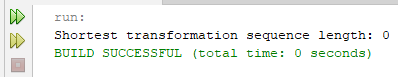
*INPUT*

**

**

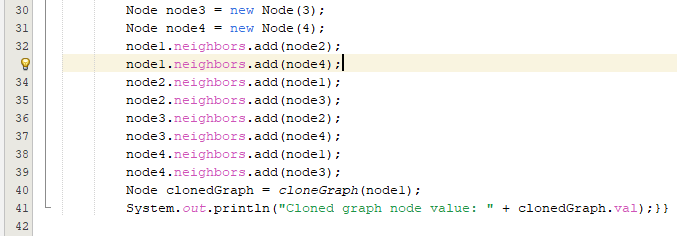
1. *Find the shortest transformation sequence from a start word to an end word, changing one letter at a time such that each transformed word exists in a given dictionary.*
2. *Use BFS to explore all possible transformations and keep track of the sequence length.*
3. *Each word must be connected by a single letter change, and BFS ensures the shortest sequence is found.*

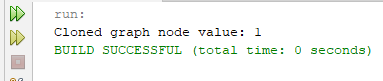
************

**

1. *Clone a graph where each node has a value and a list of its neighbors.*
2. *Use BFS to traverse and clone each node and its neighbors while maintaining the graph structure.*
3. *Ensure that all nodes are copied correctly and no node is visited more than once.*

******

**

**