Programming assignment 7.
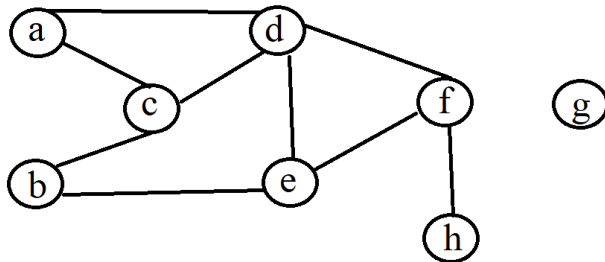
Due date: Monday, November 30, 2020 at 11:59pm

· ·

Note: you can have different names for vertices: {a b c d e ..} = {v1 v2 v3 v4 ..} = {1 2 3 4 ..}

In this program you are required to implement BFS.

First, you can create the below graph and print the resulting adjacency matrix/list. Or create a random graph.
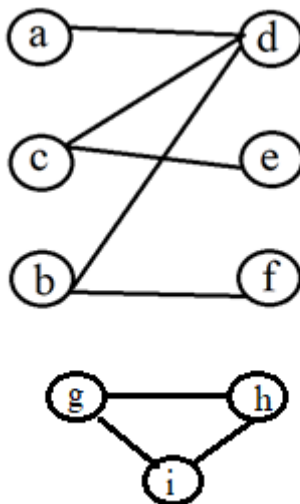

Part A.

1. Request the user to determine the starting vertex (*a*) for **BFS** algorithm
2. Call *BFS* function to find the vertices reachable from vertex *u* and print the *shortest paths* and their *lengths/distances*.

Part B.

In this part you are required to determine if a random undirected graph is <u>bipartite</u> or not. You can use the below graph to test.

Here, we work with three colors for the vertices: *gray* (not visited), [*blue, red*] (**opposite colors**)

1. Print the resulting adjacency matrix/list.

2. Implement 2 functions: *Explore* and *Is_bipartite*

3. In ***Explore*** function,

    a. For each vertex (v) initialize *v.color = "gray"*.

    b. Start from the first vertex, color it *"blue"* and call *Is_bipartite* on that.

    c. Next, go to the next unexplored vertex (having *gray* color), color it *"blue"* and call *Is_bipartite* again.

    d. Repeat *step c.* until every vertex is explored/colored or a not bipartite graph is detected.

4. Now to implement our second function (***Is_bipartite***), you need to change your BFS function in part A.

    a. Keep popping each vertex from *Q*. (call it **u**)

    b. Go to the adjacency list of **u**, (adj(u)), and for each neighbor (**v**):

    c. If ***v***. *color* == *"gray"*, assign an opposite color to **v** and push it into the *Q*. (**Example**: *u.color* is *blue*, and *v.color* is *gray* ➔ we set ***v.color* = *"red"***)

    d. Else if ***v***.*color* == ***u***.*color*: Stop the entire code and print "**NOT bipartite**".

5. Print the color of all the vertices.