Lab 07 (Part B): Adjacency Matrix

The next step is to do a **BFS** and a **DFS traversal** as shown on **lecture slides 16, Graph Traversals.**

Complete the implementation as follows:

- Functions **Graph::bfsSequence:**
  - **Parameter:** an **STL vector** of **characters**
  - Using an **STL queue**, store in the vector the **BFS** sequence (always choosing the **vertex** that comes **alphabetically first**). Sequence **starts** with vertex at **index [0]**.
  - This function is a **void** function and it is **non**-recursive**.**

- Function **Graph::dfsSequence:**
  - **Parameter:** an **STL vector** of **characters**
  - Using an **STL stack**, store in the vector the **DFS** sequence (always choosing the vertex that comes **alphabetically first**). Sequence **starts** with vertex at **index [0]**.
  - This function is a **void** function and it is **non**-recursive**.**

- Function **main**
  - Add the following code to the main function, below each graph:

```
vector<char> bfs(g1.getNumOfVertices());
g1.bfsSequence(bfs);
for (char i : bfs)
       cout << i << " ";

// Same code for DFS and repeat for other graph.
```

After printing **each graph**, your output should show:

```
BFS: (the sequence will be displayed here)

DFS: (the sequence will be displayed here)
```