Homework 6

CSE 276

Deadline: 11/25/2022

I. Shortest Path between Vertexes

Consider the following method for the `Graph` class:

```
void ShortestPath(String SourceLabel, String DestinationLabel)
```

The above method finds the shortest path from a source vertex to a destination vertex. The arguments of the method are the labels of the source vertex and the destination vertex. For finding the shortest path, the above method performs a breadth-first traversal starting at the source vertex. Accordingly, the method visits the source vertex, then visits the neighbors of the source vertex, then visits the neighbors of the neighbors of the source vertex ... . Toward this goal, the method first visits the source vertex and `enQueue` this vertex. The method `deQueue` vertexes from the Queue until the Queue becomes empty. Every time the method `deQueue` a vertex, the method checks all the neighbors of the `deQueue`-ed vertex to see which ones are not visited. The method visits those neighbors that are not visited and `enQueue` them. The method keeps a record of visited vertexes in `VisitedVertexes` array. The `ShortestPath` method has the following differences with the `depthFirstTraversal` method that we studied in previous labs:

- At the top of the body of the method we define a `PathArray`:

  ```
  String[] PathArray = new String[size];
  ```

  The `PathArray` is an array of `String`s to store shortest paths from the source vertex to other vertexes. More precisely, for any specific vertex the corresponding entry of `PathArray` is the `String` of labels of those vertexes that are on the shortest path from the source vertex to this specific vertex.
- The `PathArray` at the index corresponding to the source vertex is set to an empty `String`:

  ```
  PathArray[VertexToIndex(LabelToVertex(SourceLabel))]="";
  ```

- In `Visit`ing a vertex, the method does not print the label of the vertex. Rather, the `Visit` method records the path that is traversed from the source vertex to reach the vertex that is being visited. That path is actually the shortest path from the source vertex to the vertex that is being visited. Accordingly, the signature of the `Visit` method is as follows:

  ```
  void Visit(Vertex AnotherVertex, Vertex CurrentVertex, String[]
     PathArray)
  ```

  When `Visit` method is called, `AnotherVertex` is an unvisited neighbor of the `CurrentVertex`, and is being `Visit`ed. The shortest path from the source vertex to `CurrentVertex` is already stored in index `VertexToIndex(CurrentVertex)` of the `PathArray`.
- At the bottom of the body of the `ShortestPath` method, the following lines of code print the shortest path from the source vertex to the destination vertex:

  ```
  System.out.println("Shortest path from Vertex " + SourceLabel +"
     to Vertex "+ DestinationLabel+" is: ");

  if(PathArray[VertexToIndex(LabelToVertex(DestinationLabel))]==null
     )
  return;

  System.out.println(PathArray[VertexToIndex(LabelToVertex(
     DestinationLabel))]+DestinationLabel);
  ```

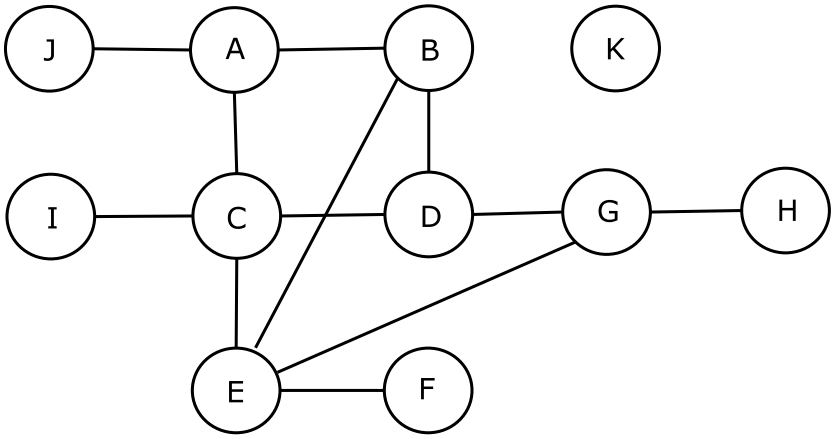
Fill in the body of the `Visit` method in the `Application.java` file. Use the following lines of code to test the developed method:

```
Graph myGraph = new Graph();
myGraph.addVertex("A");
myGraph.addVertex("B");
myGraph.addVertex("C");
myGraph.addVertex("D");
myGraph.addVertex("E");
myGraph.addVertex("F");
myGraph.addVertex("G");
myGraph.addVertex("H");
myGraph.addVertex("I");
myGraph.addVertex("J");
myGraph.addVertex("K");
myGraph.addEdge("A", "B");
myGraph.addEdge("B", "A");
myGraph.addEdge("B", "E");
myGraph.addEdge("E", "B");
myGraph.addEdge("A", "C");
myGraph.addEdge("C", "A");
myGraph.addEdge("B", "D");
myGraph.addEdge("D", "B");
myGraph.addEdge("D", "C");
myGraph.addEdge("C", "D");
myGraph.addEdge("C", "E");
myGraph.addEdge("E", "C");
myGraph.addEdge("E", "F");
myGraph.addEdge("F", "E");
myGraph.addEdge("D", "G");
myGraph.addEdge("G", "D");
myGraph.addEdge("E", "G");
myGraph.addEdge("G", "E");
myGraph.addEdge("G", "H");
myGraph.addEdge("H", "G");
myGraph.addEdge("C", "I");
myGraph.addEdge("I", "C");
myGraph.addEdge("A", "J");
myGraph.addEdge("J", "A");
myGraph.ShortestPath("A","G");
myGraph.ShortestPath("F","H");
myGraph.ShortestPath("A","K");
```

The expected output is printed below:

```
Shortest path from Vertex A to Vertex G is:
ABDG
Shortest path from Vertex F to Vertex H is:
FEGH
Shortest path from Vertex A to Vertex K is:
```

The graph that is created by the above lines of code is as follows:



II. Submitting the Assignment

When submitting your response to this assignment, keep the above lines of code in the body of the `main` method.