Lab 9

1. Show that for any simple graph G (not necessarily connected) having n vertices and m edges, if $m \geq n$, then G contains a cycle.

2. Suppose G = (V, E) is a connected simple graph. Suppose S = (V_S, E_S) and T = (V_T, E_T) are subtrees of G with no vertices in common (in other words, V_S and V_T are disjoint). Show that for any edge (x,y) in E for which x is in V_S and y is in V_T, the subgraph obtained by forming the union of S, T and the edge (x,y) (namely, U = (V_S U V_T, E_S U E_T U {(x,y)})) is also a tree.

3. Implement a subclass ShortestPathLength of BreadthFirstSearch that will provide, for any two vertices x, y in a graph G, the length of the shortest path from x to y in G, or -1 if there is no path from x to y. Use the ideas mentioned in the slides for your implementation. Be sure to add a method of the Graph class having the following signature:

   int shortestPathLength(Vertex u, Vertex v)

   which will make use of your new subclass

4. Implement a subclass ShortestPath of BreadthFirstSearch that will, for any two vertices x, y in the graph G, will return a shortest path from x to y (or null if none exists). Return type should be a List of vertices that specifies the path. Be sure to add a method of the Graph class having the following signature:

   ```
   List<Vertex> shortestPath(Vertex u, Vertex v)
   ```

   which will make use of your new subclass

5. Prove that if T is a tree with at least two vertices, T has at least two vertices having degree 1. *Hint.* Let v be any vertex in T and think of T as a rooted tree with vertex v. Create the usual levels for the tree. Then use properties of such a tree to solve the problem.