Data Structures Lab 12 Graphs Name:_____

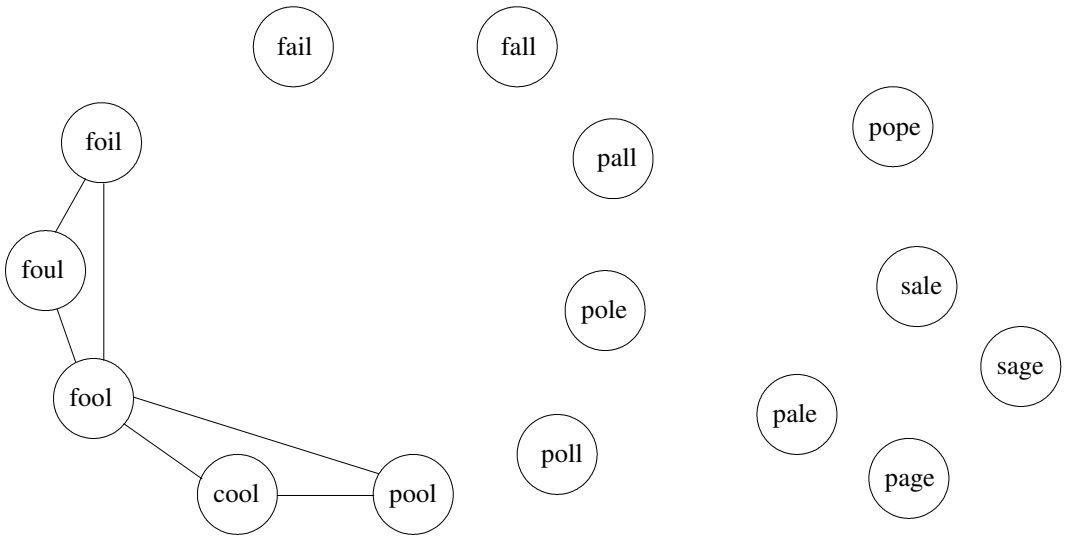
Objectives: To understand how a graph can be represented and traversed.

To start the lab: Download and unzip the file lab12.zip

Part A: In a word-ladder puzzle (discussed yesterday - see handout at: http://www.cs.uni.edu/~fienup/cs052s11/lectures/lec25_questions.pdf) you transform one word into another by changing one letter at a time, e.g., transform FOOL into SAGE by FOOL → FOIL → FAIL → FALL → PALL → PALE → SALE → SAGE.

a) We used a graph algorithm to solve this problem by constructing a graph such that
- words are represented by the vertices, and
- an edge represents?

b) For the words listed below, complete the graph by adding edges as defined above.



c) To find the shortest transformation from FOOL to SAGE, why did we decide on using a Breadth First Search (BFS) (i.e., where you find all vertices a distance 1 (directly connected) from FOOL, before finding all vertices a distance 2 from FOOL, etc) traversal?

d) From inside IDLE, run the `lab12/word_ladder.py` program. Examine the "enqueue" and "dequeue" lines of output produced by the `algorithms.bfs(g,g.getVertex("fool"))` call. Does this output match the expected "enqueues" and "dequeues" performed during a bfs of the above graph starting at "fool"?

After you have answered the above questions, raise your hand and explain your answers.

Part B: From inside IDLE, open `lab12/algorithms.py` and `lab12/graphWithPrevious.py` modules. The `algorithms.bfs` method expects to be passed a `LinkedDirectedGraphPrevious` object which includes with each vertex a `_previous`" attribute to track which vertex was visited immediately before this one in the bfs traversal.

Your task in Part B is to complete the `traversePrevious` method of the `LinkedDirectedGraphPrevious` object. The `traversePrevious(self, end)` method returns the list of labels from traversing the `_previous` attributes from '`end`' backward until `None` is reached.

After you have completed and tested your `traversePrevious` method, raise your hand and demonstrate your code by running the `word_ladder.py` program.