Stanley Lalanne
CS 312 Homework 10
Part 1,2 Questions

1)
Look at the member variables of Graph (in the Graph.h file). What representation is being used for the graph in the code provided, Adjacency Matrix or Adjacency List?

- In the Graph.h file, the variable to represent the edges between the vertices is a vector of sets of integers. std::vector<std::set<int> >
  This signifies that the graph is represented as **adjacency-list.**

Other questions:

1) The Graph class has an accessor function that returns the number of edges: int E(), and an accessor function that returns the number of vertices : int V(). However, the class has a private variable to keep track of the number of edges only (nedges), which is increased every time an edge is added to the graph. Why is it that don't we need a variable to keep track of the number of vertices? How can we quickly know the number of vertices in the graph?

- Because the vertices are added automatically when you call the add_vertex() function which only adds a new set of integers in the vector of sets. The vector increments each time a new element is added, and the vector class provides a size method that can be accessed at constant time. So to get the number of vertices, you simply need to return the size of the vector.

2) What is the big-O time for displaying the graph (the overloaded output operator)?

- To display the graph, I iterate through the vector of vertices. At each iteration is a set of integers representing the edges of a specific vertex. I then iterate through the set to retrieve all the edges.
- Iterating over the vector of vertices takes O(V) time, and iterating over the set of edges takes O(E) time. Therefore the whole procedure takes O(V*E) time.

3) What would be the big-O time for displaying the graph (overloaded output operator) if a matrix representation had been used?

- If a matrix representation has been used, the procedure would take O(V^2) time.

4. What new variables were added to LabeledGraph that were not in Graph and what do you think they are used for?

```
int nedges;
 std::vector<std::set<int> > vertices;
 std::vector<std::string> labels;
 std::map<std::string, int> indexedLabels;
```

- In the graph.h file, both the nedges and vertices variables were there. Although the vertices variable in the graph.h file was called edges. But they're the same representation.
- The labeledgraph.h file introduces the

```
 std::vector<std::string> labels;
 std::map<std::string, int> indexedLabels;
```

- The labels variable is a list of string intended to store the names of each vertex in the graph.
- The indexedLabels is a variable that maps each vertex(Label) to its position in the vector of vertices.

5. What changes do you see in the add_vertex and add_edge functions (LabeledGraph vs Graph)? Explain why you think those changes were made.

- In the Graph.h file, adding a vertex was a procedure that only adds a new set of integers to a vector. And the size of the vector was indicative of how many vertices were there, and the indices of the vector were the vertices themselves. The function took no argument.
- In the LabeledGraph.h file, the add_vertex() function takes one argument; the string or name or label of a specific vertex. That label is then added to a vector that holds all the labels. Then that label is mapped to its index in the vector of vertices.