CGL Data Structures Specification Sheet

Node Class:
Contains an ID to identify itself. Equality comparisons would check if the IDs are matching. A node precedes another if its ID is smaller than the other's. Maintains a set of edges connected to this node.

```
class Node {
        private:
                vector<Edge*> connectedEdges;
        public:
                Node(int64 id);
                int64 id;
                void addEdge(Edge* edge);
                bool operator==(const Node& other);
};
```

Edge Class:
Contains two nodes in a vector in this format {upstream node id, downstream node id}. Also keep track if it's a directed or undirected edge. If it's directed, the relationship would be upstream node->downstream node. If it's undirected, the relationship would be upstream node<->downstream node.

```
class Edge {
        private:
                vector<int64> nodes;
                bool isDirected;
                bool isBackwards;
        public:
                Edge(int64 id1, int64 id2, bool isBackwards = false);
                bool reverse();
                int64 getUpstreamId();                 // Both upstream and downstream
                                                          depends if the edge is backwards
                int64 getDownstreamId();
                int64 traverse(int id);
                bool operator==(const Edge& other); // Checks if both edges have the
                                                       same upstream node
}
```

NodeTraversal Class:
Node wrapper class that traverses the given node. Maintains a backwards value that could reverse edges. Returns any edge that would be traversed from node as the upstream node.

```
class NodeTraversal {
	private:
		Node* node;
		bool isBackwards;
	public:
		NodeTraversal(Node* node, bool isBackwards = false);
		vector<Edge*> getTraversedEdges(vector<Edge*> edges);
}
```