Homework #9 - Graphs

For the graph, rather than storing elements in the vertices, we'll just have each vertex be a unique integer. So when you make a new graph with N vertices, assume they are numbered 0 to N-1.

Level 1: Make a simple graph
Level 2: Make digraph subclass of graph
Level 3: Make a weighted digraph subclass

Graph

Swift	C++
func addEdge(firstVertex : Int, secondVertex : Int)	void addEdge(int, int)
func removeEdge(firstVertex : Int, secondVertex : Int)	void removeEdge(int, int)
func neighborsOf(vertex : Int) -> [Int]	std::vector<int> neighborsOfVertex(int)
func verticesAreAdjacent(firstVertex : Int, secondVertex : Int) -> Bool	bool verticesAreAdjacent(int, int)
func verticesAreConnected(firstVertex : Int, secondVertex : Int) -> Bool	bool verticesAreConnected(int, int)
func shortestPathFrom(_ start : Vertex, to end : Vertex) -> [Int]	std::vector<int> shortestPathFromVertex(int, int)
func hasCycle() -> Bool	bool hasCycle()

Weighted version of addEdge:

Swift	C++
func addEdge(firstVertex : Int, secondVertex : Int, withWeight weight : Double = 1)	void addEdge(int, int, double)