lab 23 Graphs via Adjacency Lists

Instructions: In this lab implement a Graph with an adjacency list.

Implement the following class:

```
#ifndef GRAPHAL_H
#define GRAPHAL_H

/* This class represents a weighted driected graph via an adjacency list.
 * Vertices are given an index, starting from 0 and ascending
 * Class W : W represent the weight that can be associacted with an edge.
 * We will not weight the vertices.
 */

template<class W>
class GraphAL {
    private:
        /* You fill out. */
    public:
        /* Initialize an empty graph. */
        GraphAL();

        /* Initialize the Graph with a fixed number of vertices. */
        GraphAL(const int vertices);

        /* Deconstructor shall free up memory */
        ~GraphAL();

        /* Adds amt vertices to the graph. */
        void addVertices(int amt);

        /* Removes a vertex.
         * return wheter sucessful or not
         */
        bool removeVertex(int idx);

        /* Adds an edge with weight W to the graph.
         * Make sure to add to the end of the list (or other functions will fail.)
         */
        bool addEdge(const int start, const int end, const W &weight);

        /*
         * Remove edge from graph.
         */
        bool removeEdge(const int start, const int end);

        void depthFirstTraversal(void (*visit)(const int node));
        void breadthFirstTraversal(void (*visit)(const int node));

        /*
         * Return adjacent weight from start to end (or -1 if they are
         * not adjacent.
         */
        W adjacent(const int start, const int end);

        /* Returns the TOTAL weight of the minimum spanning tree with the
         * given starting node.
         * You must use Prim's MST.
         */
        W prims(const int start);

        /* Print out the Graph */
        void print() const;

};

#include "graphal.cpp"

#endif
```

Write some test cases:
Create some test cases, using Unity, that you believe would cover all aspects of your code.

Memory Management:
Now that are using new, we must ensure that there is a corresponding delete to free the memory. Ensure there are no memory leaks in your code! Please run Valgrind on your tests to ensure no memory leaks!

How to turn in:
Turn in via GitHub. Ensure the file(s) are in your directory and then:

- $ git add <files>
- $ git commit
- $ git push

Due Date: November 16, 2020 2359

Teamwork: No teamwork, your work must be your own.