CMPUT 275 - Tangible Computing **Winter 2020**

Assignment #2 - Part 1 & 2: Driving Route Finder

Part 1 Rubric: Look below for the rubric for part 2. Any mention of V or E refer to the vertices or edges of the Edmonton graph (respectively).

Category	Mark
1. `I/O: Graph Loading & Input Reading` • Properly loads the Edmonton graph as a *directed graph*. • Any output produced by the algorithm is formatted *exactly* as described in the assignment specification.	/10
2. `Nearest Vertex` • Finds the vertex nearest to the start lat/lon and the vertex nearest to the end lat/lon in $O(\|V\|)$ time.	/15
3. `Dijkstra's Algorithm` • Correctly runs Dijkstra's algorithm to find a shortest driving route from the start vertex to the end vertex, or determines no such path exists. • The implementation of Dijkstra's algorithm runs in $O(\|E\| \cdot \log \|E\|)$ time using the `heap` data structure developed in the lectures.	/60
4. `Communication Protocol` • Follows the protocol exactly as described in the specification, including waiting for `A` before proceeding (when appropriate). **Note**: We do not test timeouts in part 1 (you are not expected to implement timeouts until part 2) and we will always provided "client" input according to the specification (i.e. no invalid inputs) in this part.	/15

For all grading categories, consult the assignment description for details. This is mostly giving the high-level breakdown of marks. Any other issues that do not lie directly under one of these categories but still prevent the program from functioning properly will be applied against the most relevant category at the grader's discretion.

Note: Your submission must compile simply using `make` when we extract it. Thus, you need to submit all source code files even the ones you did not modify. You are not required to submit the Edmonton graph file (we will just copy our own version into your directy), but you will not be penalized if you include it.

Further Notes: Deductions (to the overall grade) may be applied at the grader's discretion if the submission

- Does not compile due to syntax errors. Significant deductions apply if this happens, possibly resulting in a grade of 0 overall. You must submit code that can be run. *Make sure your code runs on the VM before you submit it!*
- Has extremely poor code structure, style, or modularity.

- Does not follow the submission instructions for this project.
- Does not follow the Code Submission Guidelines in any substantial way. **Good style (proper variable names, indentation, comments, etc.) is essential and should be practiced. Bad style will lead to significant mark deductions.**

Part 2 Rubric

Category	Mark
1. `Desktop Serial Port Communication` • . Requests are handled using the serial port (instead of cin/cout). Improperly formatted requests and timeouts are handled appropriately.	/25
2. `Client Serial Port Communication` • The client adheres exactly to the protocol given in the assignment description when communicating requests to the server. • Timeouts are handled properly, as are "no path" responses. • Paths with over 500 waypoints are handled safely in some manner. • Waypoints are stored properly	/50
3. `Drawing` • The route is drawn properly (when applicable).	/25

For all grading categories, consult the assignment description for details. This is mostly giving the high-level breakdown of marks. Any other issues that do not lie directly under one of these categories but still prevent the program from functioning properly will be applied against the most relevant category at the grader's discretion.

Note: The client and server should be in two separate subdirectories. Your server submission must compile simply using `make` from within its subdirectory. You are not required to submit the Edmonton graph file (we will just copy our own version into your directy), but you will not be penalized if you include it. Your client should also compile and upload to the Arduino simply using `make upload` from within its subdirectory.

Further Notes: Deductions (to the overall grade) may be applied at the grader's discretion if the submission

- Does not compile due to syntax errors. Significant deductions apply if this happens, possibly resulting in a grade of 0 overall. You must submit code that can be run. *Make sure your code runs on the VM before you submit it!*
- Has extremely poor code structure, style, or modularity.
- Does not follow the submission instructions for this project.
- Does not follow the Code Submission Guidelines in any substantial way. **Good style (proper variable names, indentation, comments, etc.) is essential and should be practiced. Bad style will lead to significant mark deductions.**