CMPUT 275 - Tangible Computing **Winter 2020**

Assignment #1 Part 1 & 2: Restaurant Finder

Part 1 Rubric: Look below for the rubric for part 2.

Category	Mark
1. `Joystick Movement` • In Mode 0, moving the joystick moves the cursor around the screen. Moving to the edge of the screen shifts the map view and displaying a new section of the map. (10 marks) • In Mode 1, moving the joystick up and down moves the highlight block in the appropriate direction, changing the highlighted restaurant (10 marks). • In either mode, pressing the joystick button switches to the other mode (5 marks).	/25
2. `Restaurant Display` • In Mode 1, 21 restaurant names are displayed on the LCD screen using the standard text multiplier of 2. The highlight block starts on the top restaurant (5 marks). • Pressing the joystick button on a highlighted restaurant consistently and correctly centres the cursor on that restaurant's location when transitioning back to Mode 0 while properly addressing the boundary cases from the assignment description (10 marks).	/15
3. `Displaying Restaurants as Dots` • Tapping on the map display visualizes all nearby restaurants (those within the map screen) as small dots (10 marks). • Tapping on the black space of the screen (off the map) should not not visualize restaurants. • We will not test button presses at the boundary between the map and the black space.	/10
4. `Displaying Names` • Insertion sort is used and is implemented properly, running in $O(n^2)$ time (15 marks). • In Mode 1, restaurant names are displayed in order from nearest to farthest by calling the insertion sort implementation (20 marks).	/35
5. `Caching` The caching idea from weekly exercise #2 is properly integrated into the project. The most recently-read block from the SD card is cached in main memory. Every time you need a restaurant, you first see if it is in the cache and only fetch it from the SD card if it is not in the cache. (15 marks)	/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: You may use other files in addition to `a1part1.cpp` if you wish (i.e if you wish to create other `.h/.cpp` files). But they must be located alongside the `Makefile` and the `a1part1.cpp` file so we can compile the entire project simply using `make`. Clearly indicate the roles of these other files in your `README` if you wish to use them.

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. `Quicksort Implementation & Use` • . Quicksort is used and properly sorts all restaurants according to their distance to the cursor.	/20
2. `Scrollable Restaurant List` • In Mode 1, scrolling below the bottom restaurant in the list displays the next 30 restaurants (5 marks). • In Mode 1, scrolling above the top restaurant in the list displays the next 30 restaurants (5 marks). • Scrolling before the 1st restaurant either does nothing or wraps around to the end. The last page of restaurants displays the correct number of restaurants (5 marks).	/15
3. `Button Column` • The right side of the display allows one to select the restaurant rating and select the sorting option using only the touchscreen. This should be done in an intuitive and easy to use manner .	/15
4. `Rating Selection` • Restaurants are only included if they have a rating above or equal to the rating threshold displayed on the rating selector button (10 marks). • Restaurants with ratings below the threshold are filtered out before sorting (ie, they are not unnecessarily included in the sort) (5 marks). • The filtering matches the threshold displayed on the rating selector button (5 marks). • Restaurants display as dots only if they are above the rating threshold (5 marks).	/35
5. `Sort Selection` • In QSORT mode, quicksort is called to sort the restaurants and in ISORT mode, insertion sort is called to sort the restaurants (5 marks). • In BOTH mode, both insertion sort and quicksort are called to sort the restaurants. Care is taken to ensure that both sorts start from the same restaurant array (ie, by rereading from the SD card in-between sorts) (10 marks). • In all three modes, the time taken in ms for each sort is displayed to the Serial monitor in the appropriate format (5 marks).	/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: You may use other files in addition to `a1part2.cpp` if you wish (i.e if you wish to create other `.h/.cpp` files). But they must be located alongside the `Makefile` and the `a1part2.cpp` file so we can compile the entire project simply using `make`. Clearly indicate the roles of these other files in your `README` if you wish to use them.

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.**