CMPT310 Assignment 1 Name: Branton Li Student#: 301311707

For my code, it requires the user to first input the file name that contains the required data for searching. Both best-first search and A\*search are implemented in the function search\_func with 1 parameter named as type. Type 0 indicates we will perform best-first search and 1 indicates that it will be A\* search. I implemented the best-first search and A\* search to run with basically the same procedure. Classes b\_node and h\_node is used for best-first search and A\*search respectively. The code will first gather all legal possible moves that are available for the current location. All the available moves will then be stored in a list and sorted according to either their heuristic value for best-first search or the cost + the heuristic value for A\* search. We will check if there are any nodes remaining in the list before popping the min value out from the list. This checking is the key to differentiate if there is a shortest path for the given map. If there are no nodes left in the list, then we will return an error value to the main to indicate no solution can be found. After the error checking, we will check if we have visited the node with the minimum value. If that node has been visited, we will keep on popping the node with the minimum value from the list until there is a node that we haven’t visited yet. We add that minimum node to the visited list and have it as our current node. An extra checking will be done after each assignment of the current node, we will check if that node is our goal and return that node if it matches. If the current node is not our goal, then we will repeat the above procedure until we find our goal, or we can conclude that there is no solution for the given data.

b\_node and h\_node are the classes that store the required info for each cell while we perform best-first search or A\* search. Both b\_node and h\_node contains the location of the cell, the heuristic value of the cell and the path that we took from the starting location to the current one. For h\_node, we have an extra attribute that stores the cost that is required to move from the starting location to the current location.

And for the heuristic function, I have the Manhattan distance as my admissive heuristic function as it matches the moving pattern that we use while searching and provides us with an accurate approximation that will not exceed that actual cost needed.