AA – Coursework- 19013163

Task 3.1 (Design Explanation):

Task 1 is implemented in two parts. The first part deals with single searches for station data while the second deals with bulk data from csv files. Part 1 is fully parallelised, it creates 4 processes, each with a different start and end point at equidistant points throughout the stations array (stored in shared memory) read in from a csv file. Each process then searches its portion of the array, it first checks perfect matches, then perfect matches on keywords and finally near matches (such as “bristl” for “Bristol”). To work out near matches the current station string is split using pythons “.split()” with each string being compared to the input string. Part 2 is fully parallelised; the processes are given two arrays (both stored in shared memory), the first array is sorted by name and the second by CRS code. The second search then loops through a list created from the bulk csv file and performs a binary search on one of the lists depending on if the input data is a station name or CRS code. Afterwards all the finished search results are output to terminal and written back to another csv file called test results. Part two uses a binary search and is therefore O(log n) time complexity whereas part one is O(log n) for the worst case as it used a parallel serial search, due to the fact there is only one input.

Task 2 is also implemented in two parts. The first part deals with minimum spanning tree between two nodes on the network. The library “networkx” is used to create a virtual adjacency matrix which is used to create the MST based on the weights of the desired edges. This virtual adjacency matrix contains 519 nodes with 696 edges, each with their own weight defined in the dictionary of edges. Part 2 of task 2 is used to find the minimum cost between all stations in the network. This function requires no input and will work out the shortest path between all stations using Dijkstra’s algorithm based off the weights of all edges. All inputs in this section are set to upper case to aid the user in finding the required stations.

Both tasks in this system are presented through an interactive terminal GUI which is fully error trapped to remove the risk of fatal errors. The terminal GUI can be restarted after any function has been performed to allow the user to choose a different function.