1806ICT Programming Fundamentals

Searching and Sorting

1. Implement a version of binary search that handles duplicates and invalid data in the sorted list.

2. Binary search requires a sorted list which is an issue when additions and deletions are also happening. Implement a data structure which has performance of order log(n) + k where n is the number of sorted entries and k is the number of additions / deletions that have been performed.

3.
   a. Implement binary search for the file *dictionary.txt*. To do this you will need to research *ftell()*, *fseek()* and *frewind()*.

   b. Using *dictionary.txt*, generate files *dictionary_n.txt* where *n* ranges from the <length of the shortest word> to the <length of the longest word> and each sorted file only contains words of length *n*. This enables a binary search for any word by first selecting the appropriate file and then, as all words are the same length, easily performing a binary search on that file. Demonstrate the performance of this implementation with that attained by the program in a) above. What are the reasons why there should be an improvement.

4. Implement an order n sort that takes advantage of the fact that all numbers are within a given limited range e.g. 100 ... 1,000. Note there may be duplicate numbers.

5. Implement a sort that splits the input array into sections, sorts each section of the input array and then merges sections to give the final sorted output.