Part 1 – Sorting and Merging

a. A strong contender for the sorting component is Merge Sort. This is because it has an efficiency of n(log n), which is good for large volumes of data and also because it already incorporates the merging of lists, meaning it could potentially be easily adapted to facilitate the merging of the multiple lists of students required in this assignment.

Another possibility is Quicksort, since it often proves to be a fast solution. However, this is not guaranteed and the worst-case scenario actually places this method’s efficiency at n2. In addition, this sorting algorithm requires more moving around of elements for each step compared to Merge Sort, meaning it could require more processing.

For merging the lists, the simplest solution would be to use the same method of merging as is used in Merge Sort, which involves adding the lowest out of the current values in the source lists to the current position in the new list and repeating until the source lists are exhausted. This will create a fully merged list ordered in a somewhat logical fashion.

Another option is to simply place the sorted lists one after another in the new list. If the start of each section were noted and accessible to a searching program, this could potentially allow increased speed when searching. However, it would not create a completely sorted list.

b. The obvious choice of data structure for this situation is a linked list. This should minimise the need to move elements around in the structure while sorting when compared to using an array. This should help to optimise the speed of the sorting. In a linked list, the data is not necessarily stored in a logical order. Instead, each element contains the address of the next element, meaning they are linked sequentially.

An array could also be considered. The main advantage of using an array would be easier access to any given element without having to follow a sequence of addresses. An array holds the data in the correct sequence, with an element’s position in memory determining its place in the array. This leads to a disadvantage in large lists, since to insert an element into the middle, all elements after it must be shifted forward.

c. For this solution, it is assumed that for student records across all colleges, the first and second names and the student number are available. The records are expected to be supplied in one list for each different college. It is also assumed that the student numbers across the different colleges can will have had a digit added which will indicate which college the student is from and also to ensure that there are no duplicate student numbers from different colleges.

The solution will use merge sort to sort the three lists separately so that they can be merged into one list easily, using the college identification digit to order them. For each list, the function will recursively split them into smaller lists until there is just one element, at which point

d.