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, especially for the large lists which would have to be handled for this assignment.

For merging the lists, one 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 (of either the student numbers or names) 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. Since this would take the form of an iterative algorithm, the efficiency would be n2.

Another option for merging is to simply place the sorted lists one after another in the new list. Since this is a very simple solution, it would be very fast, possibly with an efficiency of n. If the start of each section were noted and accessible to a searching program, it could also potentially allow increased speed when searching. However, it would not create a completely sorted list, meaning that if it were required to find an element without knowing which section it would be in, the search could take up a large amount of processing and time.

I decided to use Merge Sort for the sorting due to its efficiency and suitability for large volumes of data. For merging, I chose the solution which adds the lowest out of the current values in the source lists because even though it is not particularly fast, it should allow for a consistent level of performance when searching the resulting list.

b. There would be some advantages to using a linked list in this situation. It would minimise the need to move elements around in the lists 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 there will not be duplicate student numbers due to them coming 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 student’s surname to order them, since student number conventions will be inconsistent. When there are multiple students with the same surname, they will be sorted using their first name and any students which also have matching first names will be sorted using their student number. For each list, the function will recursively split them into smaller lists until there is just one element, at which point they will be reinserted into the previous level in a sorted manner using the merging algorithm. Once the three lists are individually sorted, the merging algorithm will be applied to them again to sort them all into one list. (Note – talk about how the algorithms work together – complexity etc)

d.

Part 2 – Searching

a.

b.