**William Carey**

**C16315253**

**Algorithms and**

**problem solving**

**Project Report**

**Course Code**

**DT-228**

**Year 1**

**Computer Science**

Contents

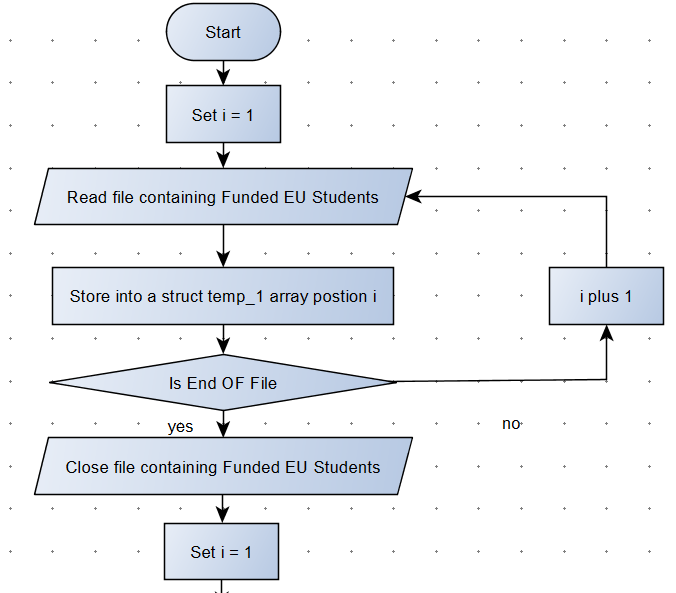
[Question 1 – Flowchart that sorts out three list into one 3](#_Toc480715868)

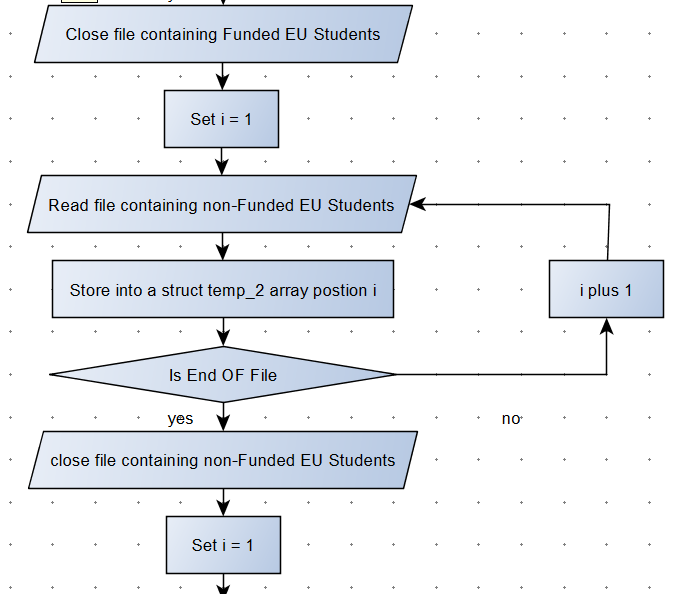
[Question 2 pseudo-code on searching for international students 10](#_Toc480715869)

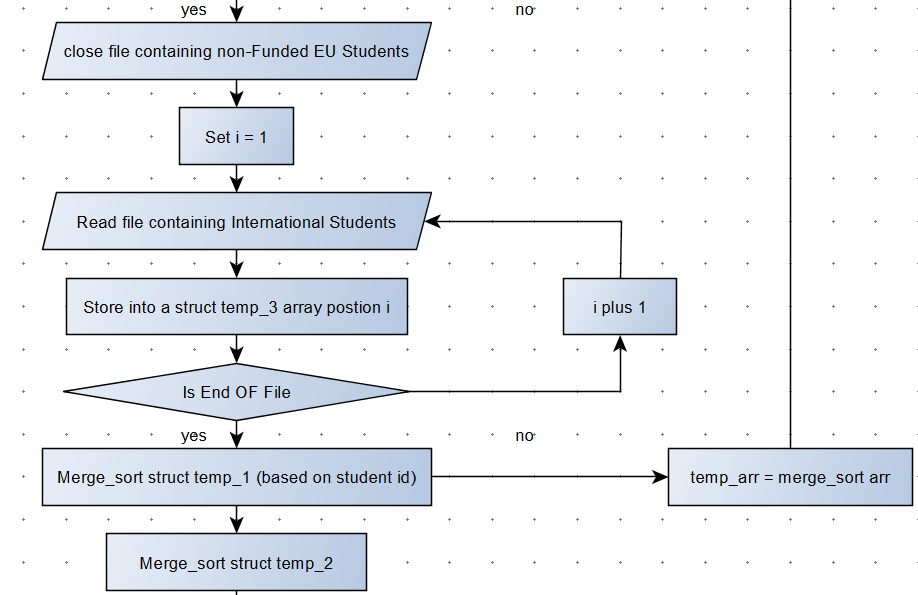
[Question 3 pseudo-code on searching for a particular student by surname 11](#_Toc480715870)

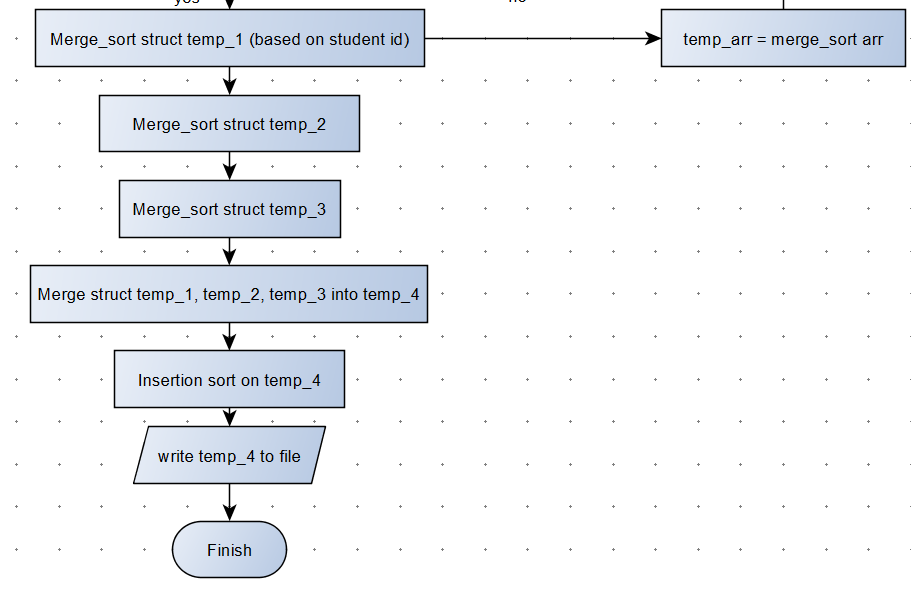
# **Question 1 – Flowchart that sorts out three list into one**

Within the merge sort, there is a insertion sort. The base case was changed to “if N < 8” to allow this change. Because of this, the big O of this algorithm is O ( (N ^ 3)(log (N))) as the insertion sort has the big O of O (N ^ 2) whereas the merge sort has a worst case of O ( N (log (N))). (N x N ^ 2) = (N ^ 3). This means within the algorithm there is a loop within a loop within a loop. All the students are sorted by student ID as that between all the student is the only thing that is unique between each student.

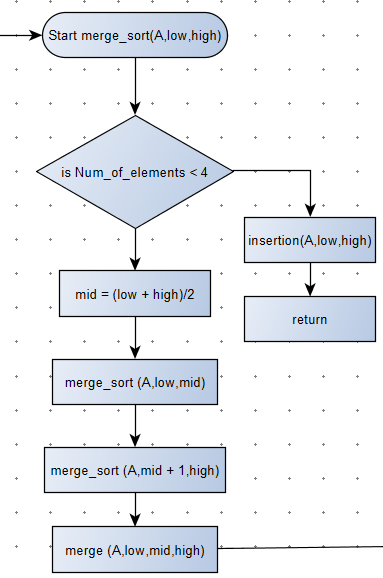


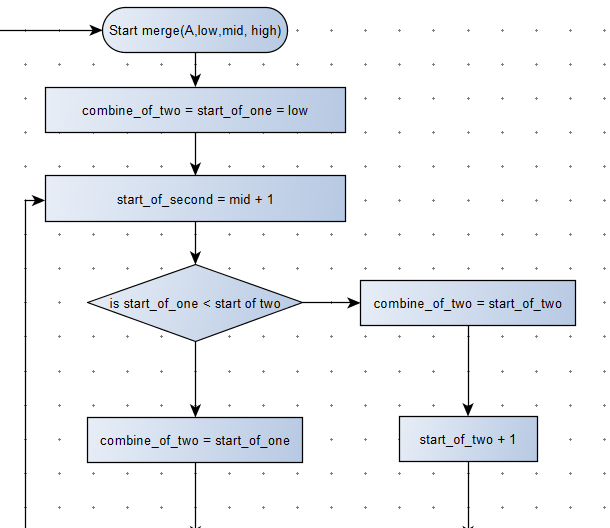


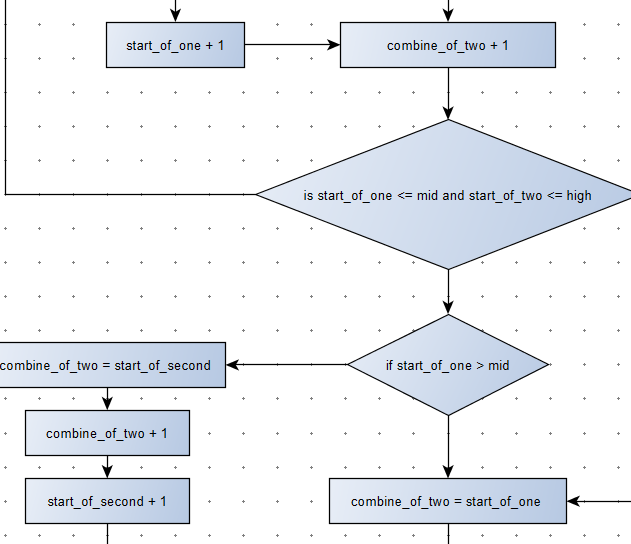


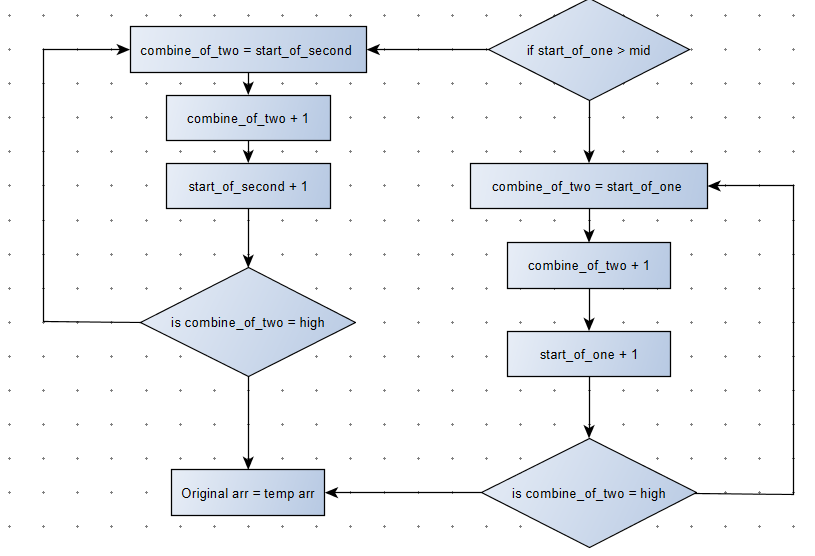


The “temp arr = merge\_sort arr” points to the merge\_sort algorithm

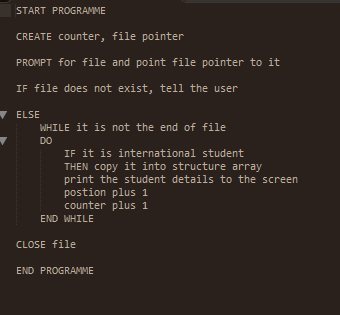








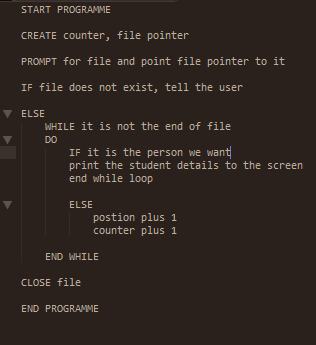
# **Question 2 pseudo-code on searching for international students**



The Big O for this algorithm is O(N) as the previous algorithm sorts the students based on student ID. So in the case of searching for international, we don’t actually know where each student is. We also know it is not sorted in student status, so we would have no idea where each international is.

There is one loop happening, which is why is has a Big O of N

# **Question 3 pseudo-code on searching for a particular student by surname**



The big O is O(N) as again we don’t have a hundred percent certain the location of the student by surname, so we must check each student linear. There is one loop to check all the students surname.