Compulsory assignment 1 – INF 102

Task 3

In the code that I have provided I use merge sort on all 4 lists, we know that merge sort has a guarantee of O(N*log(N)). After this 1 of 4 lists is looped and inside binary searched is used on 3 of the 4 lists, this checks for triplicates in the combination ABC, ABD, ACD. To check for the last combination another list is looped and binary search is again used on 3 of the 4 lists. We know that binary search has O(log(N)). And if we disregard the constants in this algorithm (ex: 4 lists -> N = 200, 1 loop = 50 elements and binary search could be searching 150 remaining elements and so on) we arrive at Big O notation: O(N*log(N).

Task 4

1. Based on my experiments the break-even S arrived at about: S = 259 (or 259 trials)
   (If I took away the cost of printing the values, this point arrived at about 535 trials)

```
BinaryPos: 5626218
BinaryPos: 2691852
BinaryPos: -1
LinearSearch-time: 1.002
BinarySearch-time: 1.002
```

2. The result (run-time) tends to fluctuate, even at the break-even point about 50% of the runs shows a faster linear search and vice versa. What one could do to get a more accurate result is to run even more trials and take the average (total time/ amount of searches). This would show a more accurate representation of what one can expect from a given "algorithm/solution".