Exam Question

1. a) pseudocode or C# code:

        // Print out all combinations of pairs of integers from the input array that when multiplied together, have a total of area.

        static void findArea(int area, int[] input)

        {

            // Sort the array first, so that we can split the array into two halves.

            // Prevents duplicates and reduces asymptotic complexity.

            Array.Sort(input);

            // Iterate over the first half of the array

            for (int i = 0; i < (input.Length / 2); i++)

            {

                // For every item in the first half of the array, iterate over every item in the second half.

                for (int j = (input.Length / 2) - 1; j < input.Length; j++)

                {

                    if (input[i] \* input[j] == area)

                    {

                        Console.WriteLine("{0}, {1}", input[i], input[j]);

                    }

                }

            }

        }

b.) Why Big-O?

Big-O notation is useful when designing a system as it shows the worst case scenario of an algorithm. This allows the system designer to decide which algorithm will be most efficient in terms of processing power, memory used and processing speed, so that users or other programs are not waiting for sorted input data. Big-O shows us how said algorithm behaves when dealing with data sets nearing infinity.

c.) Big-O notation for algorithm in part (a)

((N^2) / 4) + sorting algorithm efficiency (depends on the input size eg. nlogn) as the algorithm performs an n-squared loop, but only does that for half of the elements, therefore the total number of iterations is divided by 4.

d.) Colleague's algorithm evaluation

The colleague's algorithm uses binary search (O(log n)). However, the overall complexity depends on the sort algorithm used. For example, bubble sort would result in an overall complexity of n^2 + (n \* log n). If the sorting algorithm used was Merge Sort, the overall complexity would be n log n + (n \* log n) which is more efficient than using bubble sort.

e.) Further optimisation of colleague's code

For (int I = 0; I < input.Length; I++)

This line can be simplified by changing input.Length to (input.Length / 2). This will reduce the complexity of the loop from n^2 to (n^2) / 2.