Dheeraj Satya Sushant Viswanadham
(z5204820)

COMP3121 Assignment 1 – Question 2

2) From what the question entails, we understand that there is a one to one ratio of distinct 'n' nuts to bolts and each nut fits only one specific size of bolt. We can hence use a modified 'double quick sort' algorithm that determines whether a random bolt is too small fitting or too large fitting and then use its **matching** nut to do the same for the bolts (acting as the balance). We repeat this for each 'n' bolts, and this ultimately means that for each bolt down the line there are is an 'operation' to attempt to split this pile into two halves which would be **log(n)**. Since this is for each of the **n** distinct bolts, the **expected time complexity will be O(n) * O(log(n)) = O(n*log(n))** as required.

Example Pseudocode:

```
class Nuts {
        # … initialise …
}
class Bolts {
        # … initialise …
}
match(nut, bolt) {
        # find if correct fit or not
}
quicksort(n, b, low, high) {
        if low >= high then
                # continue as we want to go to the end of the sub-arrays
                return
        # Note – choice of pivot does not matter as everything will be sorted either way.
        index = nuts[low]

        # partition step
        p = partition(bolts, low, high, index)
        p = partition(nuts, low, high, index)

        # Recursively quicksort on the sub-arrays
        quicksort(nuts, bolts, p + 1, high)
        quicksort(nuts, bolts, low, p – 1)
}
```

Dheeraj Satya Sushant Viswanadham
(z5204820)

Partition function:

partition(A, low, high, index) {

    # Partition steps (self-explanatory) to get our piles

    # Have our match(nut, bolt) function here to help distinguish amongst individual nuts and bolts.

}

End