Q1:

In class, we have shown that Maximum-subarray problem can be solve in $O(n\ logn)$. Can you think of another divide and conquer approach that can solve the problem in $O(n)$?

(a) Complete the following pseudo code. Note that TotalSum is the sum of the current array, MaxPrefix is the maximum prefix sum of the current array, MaxSuffix is the maximum suffix sum of the current array, MaxSum is the maximum-subarray sum of the current array.

```
1: function LINEAR-FMS(A, low, high)
2:     if low = high then
3:         return (A[low], A[low], A[low], A[low])
4:     end if
5:     mid = ⌊(low + high)/2⌋
6:     Left =LINEAR-FMS(A, low, mid)
7:     Right =LINEAR-FMS(A, mid + 1, high)
8:     return LINEAR-FMCS(Left, Right)
9: end function
10:
11: function LINEAR-FMCS(Left, Right)
12:     TotalSum = Left.TotalSum + Right.TotalSum
13:     MaxPrefix = max(___, ___)//TODO
14:     MaxSuffix = max(___, ___)//TODO
15:     MaxSum = max(___, ___, ___)//TODO
16:     return (TotalSum, MaxPrefix, MaxSuffix, MaxSum)
17: end function
```

(b) What is the recurrence relation $T(n)$?

(c) Show the time complexity of the algorithm is $O(n)$.

Q2:

Consider two integers x and y, where both are 2n-bit numbers. Your task is to implement a divide and conquer algorithm to multiply these two integers. The goal is to achieve a time complexity **better than $O(n^2)$**.

(a) Design an divide and conquer algorithm to solve this problem.

(b) Analysis the time complexity of your algorithm.