Maximum Subarray Sum

Description

You are given a one-dimensional array that may contain both positive and negative integers, find the sum of a contiguous subarray of numbers that has the largest sum. For example, if the given array is {-2, -5, 6, -2, -3, 1, 5, -6}, then the maximum subarray sum is 7.

The naive method is to run two loops. The outer loop picks the beginning element, the inner loop finds the maximum possible sum with the first element picked by the outer loop and compares this maximum with the overall maximum. Finally, return the overall maximum. The time complexity of the Naive method is O(n^2). Using **Divide and Conquer** approach, we can find the maximum subarray sum in O(nLogn) time. Following is the Divide and Conquer algorithm:

1. Divide the given array into two halves
2. Return the maximum of the following three
    - Maximum subarray sum in left half (Make a recursive call)
    - Maximum subarray sum in right half (Make a recursive call)
    - Maximum subarray sum such that the subarray crosses the midpoint

Project Structure

Your code should receive input as a file with the name "**input.in**". The content of input.in file should be as follows:

- Array values separated by '.'(dot). Example input file for [-5, 10, 6, 3] -> -5.10.6.3

Your code should print the result in a file with the name "**result.out**". The content of result.out file should only consist of the integer value that your algorithm outputs. As an example for the above input sample, your output file should only include the number 19.

No.	Sample Input	Sample Output
1	-5.10.6.3	19
2	4.-5.7.8.-3.2.-4.5.-2.3.2.-6.-5	18
3	2.3.-10.3.9.2	14