Programming assignment 5.

Due date: Monday, April 5 2021 at 11:59pm

Write a recursive function to calculate the *minimum positive subsequence sum* (MPSS). In other words, of all subsequences whose sum adds to a positive number, you want to determine the minimum of such sums.

Hint:

- Use the same idea of divide and conquer algorithm for MSS, but now it is not so easy to compute $\text{MPSS}_{\text{middle}}$, (**Explain why?** (You could make a counter example on a piece of paper)

To find $\text{MPSS}_{\text{middle}}$:

1. For each subarray there are *n/2* such subsequence sums. (Find them and save them in 2 different arrays called S_L and S_R) (e.g. Let's say that the left subarray is: a_L = [2, -3, 1, 4, -6] ➔ S_L = [-2, -4, -1, -2, -6])
2. Sort S_L in *ascending* order and S_R in *descending* order.
3. Define two markers: i and j: Let i be the index marker of S_L, and j for S_R.
4. Set s_{min} = *inf*. Now start iterating through S_L and S_R:
    a. If $s = S_L(i) + S_R(j) \leq 0$, then increment i.
    b. Else if $s < s_{min}$, then set $s_{min} = s$, and increment j,
    c. Otherwise, we have $s > s_{min}$, in which case we increment j.
    d. Set $\text{MPSS}_{\text{middle}} = s_{min}$ when the elements of S_L or S_R have been exhausted.
5. **Calculate** the time complexity of your algorithm for finding MPSS on paper and show your answer to me. Running time should satisfies $T(n) = \Theta(n\log^2 n)$.
6. **Explain how/why the algorithm** for $\text{MPSS}_{\text{middle}}$ works. (You may write your answer on paper)

Ask the user for the size of the array (*n*) and generate *n* random numbers between -20 to 20.

Example:

Input: A = [2, -3, 1, 4, -6, 10, -12, 5.2, 3.6, -8],

Output: MPSS = 0.8