1. Given a list of integers S and a target number k, write a function that returns a subset of S that adds up to k. If such a subset cannot be made, then return null.
Integers can appear more than once in the list. You may assume all numbers in the list are positive.
For example, given S = [12, 1, 61, 5, 9, 2] and k = 24, return [12, 9, 2, 1] since it sums up to 24.

2. We can determine how "out of order" an array A is by counting the number of inversions it has. Two elements A[i] and A[j] form an inversion if A[i] > A[j] but i < j. That is, a smaller element appears after a larger element.
Given an array, count the number of inversions it has. Do this faster than O(N^2) time.
You may assume each element in the array is distinct.
For example, a sorted list has zero inversions. The array [2, 4, 1, 3, 5] has three inversions: (2, 1), (4, 1), and (4, 3). The array [5, 4, 3, 2, 1] has ten inversions: every distinct pair forms an inversion

3.Given a string, find the longest palindromic contiguous substring. If there are more than one with the maximum length, return any one.
For example, the longest palindromic substring of "aabcdcb" is "bcdcb". The longest palindromic substring of "bananas" is “anana".

4. Given a array of numbers representing the stock prices of a company in chronological order, write a function that calculates the maximum profit you could have made from buying and selling that stock once. You must buy before you can sell it.
For example, given [9, 11, 8, 5, 7, 10], you should return 5, since you could buy the stock at 5 dollars and sell it at 10 dollars.

5. Given an array of numbers, find the maximum sum of any contiguous subarray of the array.
For example, given the array [34, -50, 42, 14, -5, 86], the maximum sum would be 137, since we would take elements 42, 14, -5, and 86.
Given the array [-5, -1, -8, -9], the maximum sum would be 0, since we would not take any elements.
Do this in O(N) time.