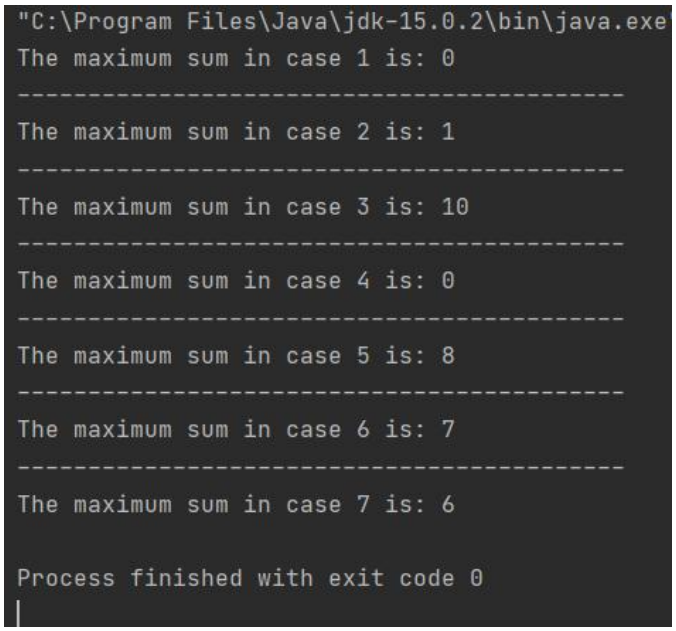
Name: Zian Wang

Test cases:



I test all the cases in one run.

Here is the algorithm:

Input: given list: x, the length of the given list: n.

Output: The maximum sum in any contiguous sublist of x.

We set sum[t] is the maximum sum of the contiguous sublist which ends in x[t]. Also, the sublist which gives the maximum sum, must ends in an element of x (Actually, any sublist of x ends in an element of x, or it is not a sublist of x). Therefore, the result we want to calculate (maximum sum in any contiguous sublist of x) is the biggest one in sum[0] to sum[n]. So if we get all the sum[t], we can know which is the result.

How we can calculate sum[t]?

There are only two possibilities: 1. sum[t] is just x[t], the other sublist ends in x[t] is smaller. 2. sum[t] is not just x[t], and because the sublist must be contiguous, sum[t] must at least have x[t] and x[t - 1]. Therefore, since we must add x[t - 1], we should add sum[t - 1] to make sum[t] maximum. So the maximum sum is the bigger one in these two possibilities. Therefore, we can get a expression: sum[t] = bigger (x[t], x[t] + sum[t - 1]). It is clear that sum[0] = x[0] (only x[0] is the contiguous sublist which ends in x[0]), so we can calculate every sum[t] start from x[0].