(I) ALGORITHM for Question 1

(start)

Step 1 : Declare global arr of 1e5 size

Declare n → size of array

sum → max sum of (+ve / 0) & initialize 0

ele_cnt → Element count & initialize 0

min → -(1e9 + 1)

// for finding Highest -ve number in [Edge case]

Step 2 : FOR i = (0) to (n - 1)

Read arr[i]

IF (arr[i] >= 0)

Add it to sum

Increase element count

ELSE

IF (arr[i] > min)

min = arr[i]

// For Highest -ve number in Array

Step 3 : IF (ele_cnt != 0) // 0 & +ve number exist

print sum, ele_cnt

ELSE

print min, 1 // If all elements are negative

I will chose Highest -ve number for max sum

(II) Dry Run for sample Input

5

1	2	-4	-2	3

	sum (0)	ele cnt (0)	min
1 >= 0	1	1	-1e9+1)
2 >= 0	3	2	-(1e9+1)
-4 ✗= 0	3	2	-4
-2 ✗= 0	3	2	-2
3 >= 0	6	3	-2

(3)

∵ ele_cnt != 0 → Ans is **6 3**

Algorithm for Question 2

Start

Step 1: Declare two global arr 'a' & 'b' of size (1e5)

Step 2: Declare test (int)
Read test
WHILE (test--) // Loop runs for each TEST
Declare n
Declare madness & initialize to 0
Read n, a[0], a[1], ... a[n-1]
b[0], b[1], .. b[n-1]

//LOGIC
FOR i = 0 to n-1
FOR j = i to n-1

IF (b[j] >= a[i])
madness = max (madness, j-1)
ELSE
break the loop
//since a & b are non decreasing sequence

print madness

(II)

Dry Run for Sample test

1
9
7 7 3 3 3 2 2 2 1
8 8 7 7 5 5 4 3 2
0 1 2 3 4 5 6 7 8

i	j	madness	j break at
0	(0 to 8) but 3	3	(j=4)
1	(1 to 8)	max(3,2)=3	(j=4)
2	(2 to 8)	max(7-2, 3) = 5	(j=8)
3	(3 to 8)	max(7-3, 5) = 5	(j=8)
4	(4 to 8)	max(7-4, 5) = 5	(j=8)
5	(5 to 8)	max(8-5, 5) = 5	"
6			
7	(6 to 8)	max(8-6, 5) = 5	"
8	(7 to 8)	max(8-7, 5) = 5	"
		max ⇒ (5)	

Final Answer ⇒ (5)