Set C Part 1

Design Marks: Total = 7

read(A,B, n)

1. read the value of *n*
2. **for** *i* ← 1 to *n*

   do read *Reg_ID*
   read *height*
   A[i] ← *Reg_ID*
   B[i] ← *height*

Evaluation criteria : **[1 mark]** Reading the values *Reg_ID* and *height* and storing it into arrays A and B respectively.

Arrange_Student(A,B, n,s)

// using two extra arrays

1. **for** i ← 0 to n-1

   do if A[i] = s

   then ht ← B[i]

   pos ← i
2. Create two arrays D and E of size *n*
3. Initialize j ← 0
4. **for** i ← 0 to n-1

   do if B[i] > ht

   then D[j] ← A[i]

   E[j] ← B[i]

   j ← j+1
5. D[j] ← s
6. E[j] ← ht
7. j ← j+1
8. final_pos ← j
9. **for** i ← 0 to n-1

do if B[i] < ht

then D[j] ← A[i]

E[j] ← B[i]

j ← j+1

10. **for** i ← 0 to n-1

do A[i] ← D[i]

B[i] ← E[i]

11. Print *final_pos* and the two arrays A and B

Evaluation criteria : **[6 marks]**	
Division:	Finding the pos of s - 1 mark
	Finding the final position of s - 2 marks
	Preserving the relative positions of students - 3 marks

Set C Part 2

Design Marks: Total = 3

read(A,B, n)

1. read the value of *n*
2. **for** *i* ← 0 to *n*-1

do read *Reg_ID*

read *height*

A[i] ← *Reg_ID*

B[i] ← *height*

3. read the value of *k*

Find_Tallest(A, B, l, r, k) *// initially l= 0, r = n-1*

// Slightly modify *Arrange_Student (A,B, n,s)* to *Arrange_Student (A, B, l, r, s)* that positions *s* between *l* and r (inclusive) and returns the value *final_pos* (1 <= *final_pos* <= *r-l+1*) of the student *s.*

1. $s \leftarrow A[l]$ // Choose an arbitrary student s in array A.

   // Here we take the leftmost student

2. $n \leftarrow r - l + 1$
3. $pos \leftarrow$ *Arrange_Student* (A, B, l, r, s)
4. $index = pos - 1$
5. **if** $index = k$

   then return A[*index*]

6. **else if** $index > k$

   then return *Find_Tallest*(*A, B, l, index - 1, k)*

7. **else return** *Find_Tallest*(*A, B, index + 1, r , k+l-index*)

Evaluation criteria : **[3 marks]**	
Division:	Modification of *Arrange_Student* ( ) - 1 mark
	Proper recursion calls of *Find_Tallestt*( )- 2 marks