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 ISBN
                 read price
                 A[$i$] ← ISBN
                 B[$i$] ← price
```

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

Locate_ISBN(A,B,n,i)

// using two extra arrays

```
1. for j ← 0 to n-1
      do if A[j] = i
              then p ← B[j]
                   loc ← j
2. Create two arrays  D and E of size $n$
3. Initialize k ← 0
4. for j ← 0 to n-1
      do if B[j] < p
                  then  D[k] ← A[j]
                        E[k] ← B[j]
                        k ← k+1
5. D[k] ← i
6. E[k] ← p
7.  k ← k+1
8. final_loc ← k
9. for j ← 0 to n-1
```

```
            do if B[j] > p
                    then D[k] ← A[j]
                         E[k] ← B[j]
                         k ← k+1
    10. for j ← 0 to n-1
            do A[j] ← D[j]
               B[j] ← E[j]
    11. Print final_loc and the two arrays A and B
```

Evaluation criteria : **[6 marks]**

Division: Finding the *price* of *i* - 1 mark
Finding the final location of *i* - 2 marks
Preserving the relative positions of books - 3 marks

Set C Part 2

Design Marks: Total = 3

read(A,B, n)

// read book details to arrays A and B

```
    1. read the value of n
    2. for i ← 0 to n-1
            do read ISBN
               read price
               A[i] ← ISBN
               B[i] ← price
```

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

// k is initialized to ceil(n/2) which is the position of the median.

// Slightly modify *Locate_ISBN(A, B, n, i)* to *Locate_ISBN (A, B, l, r, i)* that locates *i* between *l* and r (inclusive) and returns the value *final_loc* (1 <= *final_loc* <= *r-l+1*) of the book with ISBN *i.*

```
i ← A[l]                    // Choose an arbitrary book b in array A.
                            // Here we take the leftmost book
loc ← Locate_ISBN (A, B, l, r, i )
$index = loc - 1$
if $index = k$
        then return A[index]
else if $index > k$
        then return Find_Median(A, B, l, index - 1, k)
else return Find_Median( A, B, index + 1, r , k+l-index) )
```

Evaluation criteria : **[3 marks]**

Division: Modification of *Locate_ISBN*( ) - 1 mark

Proper recursion calls of *Find_Median*( )- 2 marks