Ali Mirabzadeh
305179067
CS 180 – HW6
6.19.

I use DP to solve this problem.
We begin by making an assumption that repetition of x’ of x has exactly n characters so does y’ of y . Now, let s[j] denotes the jth character of s, and s[i:j] denotes the first j character of s. Hence, we know that if s is an interleaving of x’ and y’, then its last character comes from either x’ or y’/ By removing this character, we get a smaller recursive problem on s[1:n-1] and prefixes of x’ and y’. Therefore, we consider sub-problems defined by prefixes of x’ and y’.
Let M[I,j]=yes id s[1:i+1] is interleaving of x’[1:i] and y’[1:i]

Below you can see the algorithm:

```
M[0,0]= yes
For k:1,2,…,n
    For all pairs (i,j) so that i+j=k
        If M[i-1:j]=yes and s[i+j ]= x’[i]
            M[i,j]=yes
        Else if M[i,j-1] – yes and s[i+j] – y’[j] t
            M[i,j]=yes
        Else
            M[i,j]=no
    endfor
endfor
return yes iff there is some pair (i,j) with I + j - n so that M[i,j] =yes
```

Ali Mirabzadeh
305179067
CS 180 – HW6
5.2

Based on the algorithm provided in the book for $i < j$ , $a_i > a_j$

For this question I modify the provided algorithm so that it would satisfy the requirement

```
Merge-and-Count(A,B)
Maintain a Current pointer into each list, initialized to point to the front elements
Maintain a variable Count for the number of inversions, initialized to 0
While both lists are nonempty:
    Let $a_i$ and $b_j$ be the elements pointed to by the Current pointer
    Append the smaller of these two to the output list
    If $b_j$ is the smaller than twice the element then
        Increment Count by the number of elements remaining in A
    Endif
    Advance the Current pointer in the list from which the smaller element was selected.
EndWhile
Once one list is empty, append the remainder of the other list to the output
Return Count and the merged list
```

Ali Mirabzadeh
305179067
CS 180 – HW6
5.3

Create two sets S1 and S2 from the original set in which each has n/2 elements

Below is the algorithm:

```
If S=1 return the card
If S=2
    Check if two cards are equivalent
    Return either of the cards
Endif
Call the function recursively for S1
If a card is returned
    Then test it against all other cards
Endif
If no card with the desired requirement has not yet been found
    Call the function with S2
    If a card is returned
        Test it against all other cards
    endif
endif
return a card, representing the class of equivalent cards, if found it
```

Ali Mirabzadeh
305179067
CS 180 – HW6
5.5

We sort the lines in an increasing order of their slopes

We use divide and conquer algorithm

Our base case would be for three lines.

First, we recursively compute the sequence of visible lines among all the sorted lines. Also we compute all the intersection between two lines in which will be in an increasing order by their x-coordination. For if two lines are visible, the region in which the line of smaller slope is uppermost lies to the left of the region in which the line if larger slope is uppermost.

We keep recursively doing this till get to the base case and compare the regions based on x-coordination. If all regions of a line lies underneath other lines then that line is invisible, so we remove that line from our sorted list.

At the end, return the list in which contains all the lines in which some portion of them are visible.