CSE 6331 Homework 4
Due: Thursday, February 1, by class time

Note: This homework carries double weight in terms of workload and credit.

1. Write a recursive, divide-and-conquer algorithm Power(a, n) that computes the number $a^n$, where $a, n$ are positive integers. Analyze your algorithm. Your algorithm must work in $o(n)$ time.

2. Rewrite your algorithm Power(a, n) as a non-recursive (iterative) one. (The running time must still be $o(n)$.)

3. Consider the closest-pair algorithm. Suppose we do not sort $A[i..j]$ by y-coordinate in Closest-Pair($A[i..j], (p, q), ptr$), but instead we sort the whole set of $n$ points (i.e., $A[1..n]$) by y-coordinate into a linked list in the beginning of the algorithm, immediately after sorting them by $x$. (The procedure Closest-Between-Two-Sets remains intact, but the linked list pointed to by $ptr$ now contains the entire set of $n$ points.) Does the modified algorithm work correctly? Justify your answer. (If YES, explain why; if NO, give a counterexample.)

4. Whether the above algorithm is correct or not, what is its time complexity?

5. Let $A[1..n]$ and $B[1..n]$ be two arrays of integers, each sorted in nondecreasing order. Write a divide-and-conquer algorithm that finds the $n$th smallest of the $2n$ combined elements. Your algorithm must run in $O(\log n)$ time. You may assume that all the $2n$ elements are distinct. (Write your algorithm in pseudo-code and explain it in plain English.) Note: The input consists of two arrays of the same size. When you divide the problem, make sure that the two (sub)arrays of each subproblem are of equal size.

Hint:

```plaintext
function n-smallest($i_a, j_a, i_b, j_b$)
   //The $n$th smallest element is in $A[i_a..j_a] \cup B[i_b..j_b]$/
   //You should always keep $i_a - j_a = i_b - j_b$/
   global array $A[1..n], B[1..n]$

   if $i_a = j_a$ and $i_b = j_b$
      return ________________________________  //return $A[i_a]$ or $B[i_b]$/
   else
      $m_a \leftarrow ________________________________  //i_a \leq m_a \leq j_a$/
      $m_b \leftarrow ________________________________  //i_b \leq m_b \leq j_b$/

      if $A[m_a] < B[m_b]$
         return n-smallest( ) //recursively call n-smallest/
      else
         return n-smallest( ) //recursively call n-smallest/

Initial call: n-smallest(1, $n, 1, n$)
```