Due Date: October 4, 2011

Problem 1: [10pts] Let \( I \) be a set of \( n \) intervals on \( \mathbb{R}^1 \). Show that \( I \) can be preprocessed into a data structure of size \( O(n \log n) \) so that all intervals of \( I \) that are completely contained in query interval \( \gamma \) can be reported in \( O(\log n + k) \) time.

Problem 2: [10pts] Let \( \mathcal{R} \) be a set of \( n \) rectangles in \( \mathbb{R}^2 \). Show that \( \mathcal{R} \) can be preprocessed into a data structure of size \( O(n \log n) \) so that all \( k \) rectangles of \( \mathcal{R} \) containing a query point \( p \) can be reported in \( O(\log^2 n + k) \) time. Improve the running time to \( O(\log n + k) \). (Hint: Construct a segment tree on the \( x \)-intervals of the rectangles and store a secondary data structure at each node of the tree.)

Problem 3: [10pts] Let \( S \) be a set of \( n \) points in \( \mathbb{R}^2 \). Show that \( S \) can be preprocessed into a data structure of size \( O(n \log^2 n) \) so that all \( k \) points of \( S \) lying in a query equilateral triangle can be reported in \( O(\log^2 n + k) \) time. (Hint: Construct a multi-level range tree.)

Problem 4: [10pts] Let \( S \) be a set of \( n \) pairwise disjoint segments in \( \mathbb{R}^2 \), and let \( p \) be a point not lying on any segment of \( S \). Describe an \( O(n \log n) \) algorithm for computing all segments that are visible from \( p \) — a segment \( e \) is visible from \( p \) if there is a point \( q \in e \) such that the segment \( pq \) does not intersect the interior of any other segment.

Problem 5: [15pts] (i) Given a collection \( \mathcal{R} \) of “red” nonintersecting line segments and another collection \( \mathcal{B} \) of “blue” nonintersecting segments in \( \mathbb{R}^2 \), show that all red-blue intersections (intersections between a red segment and a blue segment) can be counted in time \( O(n \log^2 n) \), where \( n = |\mathcal{R}| + |\mathcal{B}| \). What is the space complexity of your algorithm? Improve the space complexity to \( O(n) \).
   (Hint: Use a segment tree on \( x \)-projections of segments and, at each node \( v \), count intersections among the segments stored at \( v \). You need to store some additional information at each node of the segment tree. Make sure that each intersection is counted exactly once.)
   Extra credit: Improve the time complexity to \( O(n \log n) \).