CSc 437
Homework 1 (100 pts.)
Due: 9/20/07

Instructions. All assignments are to be completed on separate paper. Use only one side of the paper. Assignments will be due at the beginning of class, or via email. To receive full credit, you must show all of your work.

All questions are taken from the textbook

1. 1.1

2. 1.6b

   Answer: *We say that a vertex is* reflex *the if the inner angle it defines (inside the polygon) is larger than π. The idea is to delete vertices from P, which are clearly not vertices of $CH(P)$, namely reflex vertices. When we delete a vertex v from P, we replace the two edges connecting v to its neighbors by a single edge connecting the neighbors. This might caused one or both of the neighbors to became reflex. During the algorithm we maintain a list L of the reflex vertices of P. At each iteration, we pick a reflex vertex v from L, delete it, and insert the neighbor(s) of v if it became reflex. Initially L contains all the reflex vertices of P. The algorithm terminates when L is empty, and which point P is empty. Clearly the running time is $O(n)$.*

   The decision which vertex of L to delete is arbitrary, but the proof of correctness of the algorithm is simpler if we pick the leftmost one. In this case, the polygon maintain simple (i.e. does not cross itself). This property does not hold if we pick arbitrary vertex (give an example).

3. 1.8

4. You are given pointers to two arrays in the memory of your computer. One points to an array array $A[1..n]$ containing the vertices of a convex polygon P, and the array $B[1..n]$ containing the vertices of a convex polygon Q. Suggest an algorithm that finds in time $O(\log^2 n)$ if P and Q has a point in common, (actually doable in $O(\log n)$ time).

   Answer:

   We first find the highest and lowest point on each polygon — can easily be done in $O(\log n)$ using a binary search (note that some caution need to be taken because the vertices are in a cyclic order). This divides each polygon into two y-monotone chains. The main loop searches for vertices of P that lie inside Q. This process is repeated for each of the 4 chains (if needed), refining the search obtain for the previous search.

(a) *Pick one of these chains. Call it L, and assume WLOG that it belongs to P. Let v be a vertex of L, and let ℓ be the horizontal line passing through v. The main loop of our algorithms performs binary search on the vertices of L, until we find the two closest one such that $P \cap Q$, if not empty, is above one vertex and below the other, or determine that $P \cap Q = \emptyset$. We would then turn our attention to one of the other chains etc.*

   i. *We next check if ℓ is fully above or below ℓ. If Q is fully above ℓ, we replace v with a vertex of L above v, and continue. This is justified, since if P and Q intersect, the intersection region must lie above ℓ. The case that ℓ is fully below ℓ is treated symmetrically.*
   Note that for an edge e of one of the other chains, we can define in constant time whether e is fully above, fully below, or intersects ℓ. Hence after $O(\log n)$ time we can find the intersection points of ℓ with all chains. Let e_l, e_r be the edge of intersection of ℓ with P, (where e_l is to the left of e_r), and f_l, f_r be the edge of intersection of ℓ and Q. We sort their intersection points along ℓ.
   Of course, if e_l if to the right of f_r but e_r it to the right of f_l, then from convexity there is an intersection point of ℓ, and we are done. The symmetric case is handles similarly. So assume that $P \cap \ell$ is to the right of $Q \cap \ell$. In this case, consider the lines passing through e_r (call it ℓ_P) and f_l (call it ℓ_Q). Note that P is fully to the left of ℓ_P and that Q is to the right of ℓ_Q). If these lines do not intersect, then P and Q cannot intersect. Otherwise the only side that P and Q might *intersect is the side of ℓ that ℓ_P and ℓ_Q intersect, and the next vertex of v that we check is on this side.*

5. Show that for the line sweep algorithm to be correct, it is enough to maintain for every segment what is the next event that happens along this segment. Note that if this all you remember, the proof of the correctness of the algorithm need to be revised. How would you use this fact to guarantee that the space requirement of the algorithm to be $O(n)$.

6. You are given a polygon P with n vertices $\{v_1 \dots v_n\}$ in the order along the polygon. Also given a set S of n points. Find for each point of S if it lies inside P. The total time of your algorithm should be $O(n \log n)$.

   Answer: *We perform a line-sweep algorithm on the segments forming the boundary of P, as described in class. The events are all endpoints of segments, and all points of S. While doing so, each segment s "maintains" whether P is locally above or below s. When a point of S is meet, we check from the status what is the segment s just below it, and whether this point lies on the same side of s as P.*

7. 2.5
8. 2.8
9. 2.10
10. 2.11

    Answer: *We replace each circle with 2 semi-circles, the lower semi-circle and the upper one. Their endpoints replace the endpoints of the segment in the "standard" line sweep. We also change the code the code so it finds if two semi-circles intersect, and where. This is the only modification needs to be dons.*

11. 2.14