CS3451. Fall 08. **Solution to Q02** Grade: out of 20

Write legibly. Use points, vectors and operators discussed in class, such as n(), U(), V(), R(), +,–,• .i, b, .k... You may use shortcut pseudocode constructions such as "for (each edge (A,B) of polygon P) if…"

---

1) <3 points> The topological operators .i, .b, .k… discussed in class should be interpreted with respect to the set **R** of real numbers. Let S = [1] +]2,3[+]3,4]. Write (in simplest form) of the following sets:

S.b = [1] + [2] + [3] + [4]

S.i =]2,3[+]3,4[

S.r = [2,4]

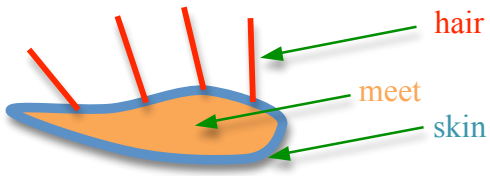
---

2) <3 points> Provide topological definitions of the hair, skin, meet parts of the closed set S shown below.

hair = S – S.r

meet = S.i

skin = S.i.b



---

3) <2 points> Let B be a ball and T a table. Both are regularized sets. Provide a topological condition that characterizes physically plausible arrangements where the ball touches the table, without interfering with it.

(B.i ∩ T.i ==∅) && (B ∩ T !=∅)

Other formulations may also be correct, for instance: (B.i ∩ T.i ==∅) && (B.b ∩ T.b !=∅)

---

4) <1 points> Provide the code for testing whether the polygonal path {A,B,C} turns towards the right at B.

boolean right(pt A, pt B, pt C) { return R(AB)•BC>0 }

Also acceptable are: R(AB)•AC>0 and dot(R(V(A,B)),V(A,C))>0…

---

5) <3 points> Points A, B, C are in the plane. Provide the code for testing whether point B lies inside the closed edge(A,C)

boolean PinE(pt A, pt B, pt C) { return (R(AC)•AB==0) && (BA•BC<=0) }

---

6) <2 points> Points Q, R, A, and B, are in the plane. Provide the code for testing whether edge(Q,R) crosses edge(A,B).

boolean ExE(pt Q, pt R, pt A, pt B) {return (right(Q,R,A)!=right(Q,R,B))&&(right(A,B,Q)!=right(A,B,R))}

---

7) <2 points> Provide the algorithm for finding a "clean" point R such that edge(Q,R) does not hit any vertex of polygon P. You may assume that P fits inside a disk of radius r.

```
pt clean(pt Q, poly P, float r) {pt R=pt(0,0);  // initialization
  repeat {        R=pt(r,random(r)*PI;   // keep picking random point R until you find a clean ray (infinite loop?)
                  boolean hit=false;      // used to track whether ray hits any vertex
                  for (each vertex V of P) if PinE(Q,V,R) hit=true;  // hit is set if any vertex is hit
                  if(!hit) return R;   }  // if no vertex was hit: return R (we expect that this will happen after one or a few trials!
  }
```

---

8) <4 points> Provide the algorithm for testing whether point Q is inside polygon P. You may use primitive operations, such as n(), U(), V(), R(), and also the right(), PinE(), and Exe() functions defined above. If you need other functions, please provide their code. You may assume that P fits inside a disk of radius r and that Q is not in P.b.

```
boolean PinP(pt Q, poly P, float r) {
        boolean in = false;          // parity toggle
        pt R = clean(Q,P,r);     // finds R outside of P such that Edge(Q,R) does not hit any vertex of P
        for (each edge (A,B) of P) if ExE(Q,R,A,B) in=!in;  // toggle when ray intersects edge(A,B)
        return in; }
```