Programming Assignment #7

CS 200, Fall 2015

Due Thursday, November 5

1. Implement a package for manipulating half–planes. I will supply you with the header file `HalfPlane.h`, which contains the following declarations, the details of which are spelled out below.

```
float dot(const Hcoords& h, const Point& Q);

Hcoords HalfPlane(const Vector& n, const Point& C);
Hcoords HalfPlane(const Point& A, const Point& B, const Point& P);

struct Interval {
  float bgn, end;
  Interval(float a=0, float b=1) : bgn(a), end(b) { }
  bool IsEmpty(void) const { return bgn > end; }
};

Interval ClipSegment(const Hcoords& h, const Point& P, const Point& Q);
```

(the `Affine.h`) header files has been included). You are to implement the items in this package.

`dot(h,Q)` — computes the dot product of the half–plane h, which specified by its homogeneous coordinate representation, and the point Q.

`HalfPlane(n,C)` — returns the homogeneous coordinate representation of the half–plane h with outwardly pointing surface normal vector $\vec{n}$, and whose bounary line ∂h contains the point C.

`HalfPlane(A,B,P)` — computes the homogeneous coordinate representation of the half–plane h whose boundary ∂h passes through the points A and B, and whose *interior* contains the point P. Note that that return value h should be such that $h \cdot A = 0$, $h \cdot B = 0$, and $h \cdot P < 0$. You are to assume that the points A, B, P are non–colinear.

`ClipSegment(h,P,Q)` — computes the intersection interval $I = [a, b]$ that corresponds to the intersection of the half–plane h and the line segment $\overline{PQ}$ with endpoints P, Q. If the intersection is empty, then $I = \emptyset$; i.e., $a > b$. If the I is not empty, then the intersection of h and $\overline{PQ}$ is the line segment $\overline{P'Q'}$, where $P' = P + a(Q - P)$ and $Q' = P + b(Q - P)$.

Your submission for this portion of the assignment should consist of a single implementation file, named `HalfPlane.cpp`. You may only include the `HalfPlane.h` and `Affine.h` header files.

2. The header file `PointContainment.h` declares the two function prototypes

```
bool PointInTriangle(const Point& P, const Point& A,
                     const Point& B, const Point& C);

bool PointInMesh(const Point& P, Mesh& mesh);
```

(the header files `Affine.h` and `Mesh.h` have been included).

`PointInTriangle(P,A,B,C)` — returns *true* if the point P is inside of the triangle with vertices A, B, and C. It returns *false* if P is outside of the triangle. It is assumed that the points A, B, C are non–colinear.

`PointInMesh(P,mesh)` — returns *true* if the point P is inside of the specified mesh, and returns *false* if P is outside of the mesh. The point P is assumed to be in *object coordinates*. To be efficient, you should first do a simple bounding box rejection test: if P lies outside of the bounding box for the mesh, simply return *false*. Otherwise, you will do a more refined test to determine if P actually lies inside of the mesh.

For this part of the assignment, you should submit a single implementation file named `PointContainment.cpp`. You may only include the header files `Affine.h`, `HalfPlane.h`, `Mesh.h`, and `PointContainment.h`.