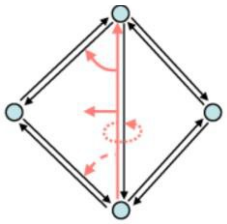
CS 418: Interactive Computer Graphics
In-class Worksheet

NETIDS:

Subdivision Surfaces

1. The half-edge data structure is designed to be efficient for the neighborhood queries required by subdivision. It stores a *vertex array*, *face array*, and *half-edge array* holding objects of the types shown in the C style declarations below.

```
struct HE_edge{

      HE_vert*    end;          // vertex at the end of the half-edge
      HE_edge* opposite;    // oppositely oriented adjacent half-edge
      HE_face*  leftFace;    // face the half-edge borders
      HE_edge* next;         // next half-edge around the face
};

struct HE_vert{

      float x, y, z;
      HE_edge* edge;   // one of the half-edges emanating from the vertex
};

struct HE_face{

      HE_edge* edge;  // one of the half-edges bordering the face
};
```



How would you implement a function
`HE_vert * start(HE_edge * e)` that returns the starting vertex of a half-edge?

2. How would you implement a function computes the average position of all neighbors of a vertex V? Call it `void nbr_avg(HE_vert *V)` Use pseudo-code…you don't have to write syntactically correct C++.

3. In the average case, how many operations are required to find all the neighbors of a given vertex using a half-edge data structure? You can assume the mesh is a closed manifold triangle mesh for which the Euler characteristic V-E+F=2 is true. How does that compare to using an indexed face set data structure?