1. Implementing Subdivision Surfaces

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.

```c
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
};
```

In the average case, how many operations are required to find all the neighbors of a given vertex using a half-edge data structure? How does that compare to using an indexed face set data structure? Assume each vertex has valence 6.

How would you implement a function

```c
HE_vert * start(HE_edge * e) that returns the starting vertex of a half-edge?
```
2. **Physics Engine**

A particle begins at

\[
\begin{bmatrix}
1 \\
2 \\
3
\end{bmatrix}
\]

and is moving with velocity

\[
\begin{bmatrix}
1 \\
-1 \\
2
\end{bmatrix}
\] per second,

and acceleration

\[
\begin{bmatrix}
0 \\
1 \\
-1
\end{bmatrix}
\] per second per second.

**a.** Use the second integral of the acceleration to compute the position after 5 seconds. The update equation you should use is:

\[
p' = p + \dot{p}t + \ddot{p} \frac{t^2}{2}
\]

**b.** Calculate the position using 5 time steps of 1 second each using the update equations below.

\[
p' = p + \dot{p}t \\
\dot{p}' = \dot{p} + \ddot{p}t
\]

**c.** What is the error? Explain why it happens.