Game Engineering 2D
Final example problems
Fall 2010

1. Look back at any of the transformation between spaces problems, C# programing problems, and Quadtree problems from Midterm 1 (final is cumulative)

2. Given the following C# Program:

```
class Increment
{
    public mCounter = 0;
    public void Increment()
    {
       for (int i = 0; i < 100; i++)
       {
           mCounter = mCounter + 1;
       }
    }
}
class Test
{
    public static void Main(string[] args)
    {
        Increment inc = new Increment();
        Thread t1 = new Thread(inc.Increment);
        Thread t2 = new Thread(inc.Increment);
        Thread t3 = new Thread(inc.Increment);
        Thread t4 = new Thread(inc.Increment);
        Thread t5 = new Thread(inc.Increment);

        t1.Join(); t2.Join(); t3.Join(); t4.Join(); t5.Join();

        Console.WriteLine("Counter Value = " + inc.mCounter);
    }
}
```

   (a) What is the smallest value that could be printed out?

   (b) What is the largest value that could be printed out?

   (c) Modify the code so that the value printed out is deterministic

3. Given the following code to be run per-thread:

```
class Compute
{
   Vector2[] mPositions;
   Vector2[] mVelocities;
   int mLow;
   int mHigh;
   int mTotal;

   public Compute(Vector2[] positions, Vector2[] velocities, int low, int high, int t
   {
       mPositions = positions;
       mVelocities = velocities;
       mLow = low;
       mHigh = high;
       mTotal = total;
   }

   public void ComputeOne()
   {
      for (int i = mLow; i < mHigh; i++)
      {
         for (int j = 0; j < mTotal; j++)
         {
            if (i != j)
            {
               lock (mPositions)
               {
                  Vector2 delta = mPositions[j] - mPositions[i];
                  float length = delta.Length();
                  delta.Normalize();
                  mVelocities[i] += delta * 0.001f / length * length;
               }
            }
         }
         lock (mPositions)
         {
            mPositions[i] += mVelocities[i];
         }
      }
   }
}
```

and the following update loop:

```
protected override void Update(GameTime gameTime)
```

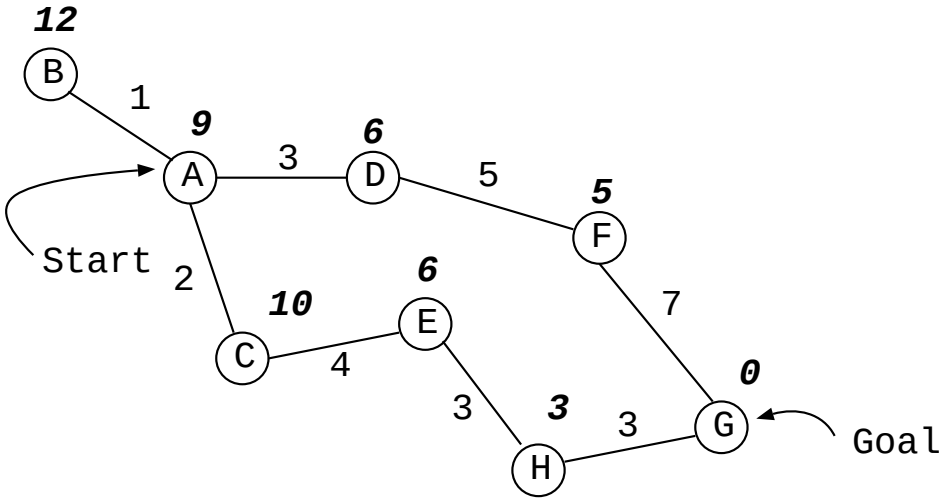
```
{
   int particlesPerThread = NumParticles / NumThreads;
   for (int i = 0; i < NumThreads; i++)
   {
      mComputation[i] = new Compute(mPositions, mVelocities,
                                    i*particlesPerThread,
                                    (i+1)*particlesPerThread, NumParticles);
      mThreads[i] = new Thread(mComputation[i].ComputeOne);
      mThreads[i].Start();
   }

   for (int i = 0; i < NumThreads; i++)
   {
      mThreads[i].Join();
   }
   base.Update(gameTime);
}
```

(a) How would the performance of this program change if we changed NumThreads from 1 to 4 (assuming that our machine had at least 4 processors)?

(b) What could we do to improve performance?

4. Given the following graph (Heuristic h() values in bold italic):



(a) Is h() admissible?

(b) Show the order that nodes would be expanded by Uniform Cost Search

(c) Show the order that nodes would be expanded by A*

(d) Show the order that nodes would be expanded by greedy

(e) Show the path that A* would compute

(f) Show the path that greedy would compute