6.189 Multicore Programming Primer **Mini-Quiz #2 (1/12/2007)**

Name: ____

American Idol has changed its voting method as follows: Before voting begins, each contestant is given 1,000 points. When a voter calls in, he or she has to answer the question "Contestant A was better than contestant B" by selecting two contestants A and B. Then 1% of B's points are deducted from him and given to A.

You have been appointed the election software guru of American Idol. As millions of people call-in to vote, you need a parallel/concurrent system for vote collection.

```
public class Contestant {
     static int currId = 0;
     String name;
     int points;
     int id;

     Contestant() {
          this.points = 1000;
          this.id = currId++;
     }
     …
     public void AddPoints(int points) {
          this.points += points;
     }
     public int GetPoints( ) {
          return this.points;
     }
     public int getId( ) {
          return this.id;
     }
}

public class Election extends Thread  {
     static final int numcan = 10;
     static final int numBots = 256;
     Contestant can[numCan] = new Contestant[numCan];
     VoteCountBot counters[numBots];

     Public static void main(String[] args) {
          for(int i = 0; i < numBots; i++) {
               counters[i] = new VoteCountBot(this, can, …);
               counters[i].start();
          }
     }
}
```

```
public class VoteCountBot {
      Election elect;
      Contestants can[];

      VoteCountBot(Election, elect, Contestants can[], …) {
            this.elect = elect;
            this.can = can;
            …
      }

      public void run( ) {
            while(true) {
                  int canBetter;
                  int canThan;
                  …
                  movePoints(can[canBetter], can[canThan]);
            }
      }
}
```

Assume that on the first day of the contest all the candidates were uniformly bad. Even though millions voted, they were all family, friends and local townies. Thus all the votes were uniformly distributed across all the candidates.

Explain how the election was affected for the following five implementations of `movePoints`. Which one would you choose and why?

```
void movePoints(Contestant canBetter, Contestant canThan) {
      int tmp;
      synchronized(canBetter) {
            synchronized(canThan) {
                  tmp = canThan.GetPoints()/100;
                  canThan.AddPoints(-tmp);
                  canBetter.AddPoints(tmp);
            }
      }
}
```

```
void movePoints(Contestant canBetter, Contestant canThan) {
      int tmp;
      Contestant first, second;
      first = (canBetter.GetId()>canThen.GetId())?CanBetter:CanThen;
      second = (canBetter.GetId()>canThen.GetId())?CanThan:CanBetter;
      synchronized(first) {
            synchronized(second) {
                  tmp = canThan.GetPoints()/100;
                  canThan.AddPoints(-tmp);
                  canBetter.AddPoints(tmp);
            }
      }
}
```

```
void movePoints(Contestant canBetter, Contestant canThan) {
      int tmp;
      synchronized(elect) {
            tmp = canThan.GetPoints()/100;
            canThan.AddPoints(-tmp);
            canBetter.AddPoints(tmp);
      }
}
```

```
void movePoints(Contestant canBetter, Contestant canThan) {
     int tmp;
     synchronized(canThan) {
          tmp = canThan.GetPoints()/100;
          canThan.AddPoints(-tmp);
     }
     synchronized(canBetter) {
          canBetter.AddPoints(tmp);
     }

}
```

```
void movePoints(Contestant canBetter, Contestant canThan) {
     int tmp;
     synchronized(canThan) {
          tmp = canThan.GetPoints()/100;
     }
     synchronized(canThan) {
          canThan.AddPoints(-tmp);
     }
     synchronized(canBetter) {
          canBetter.AddPoints(tmp);
     }

}
```