3. This question involves reasoning about the code from the GridWorld case study. A copy of the code is provided as part of this exam.

   An opossum is an animal whose defense is to pretend to be dead. The `OpossumCritter` class, shown below, will be used to represent the opossum in the grid. An `OpossumCritter` classifies its neighbors as friends, foes, or neither. It is possible that a neighbor is neither a friend nor a foe; however, no neighbor is both a friend and a foe. If the `OpossumCritter` has more foes than friends surrounding it, it will simulate playing dead by changing its color to black and remaining in the same location. Otherwise, it will behave like a `Critter`. If the `OpossumCritter` plays dead for three consecutive steps, it is removed from the grid.

   You will implement two of the methods in the following `OpossumCritter` class.

```
public class OpossumCritter extends Critter
{
  private int numStepsDead;

  public OpossumCritter()
  {
    numStepsDead = 0;
    setColor(Color.ORANGE);
  }

  /** Whenever actors contains more foes than friends, this OpossumCritter plays dead.
   *  Postcondition: (1) The state of all actors in the grid other than this critter and the
   *  elements of actors is unchanged. (2) The location of this critter is unchanged.
   *  @param actors a group of actors to be processed
   */
  public void processActors(ArrayList<Actor> actors)
  {  /* to be implemented in part (a) */  }


  /** Selects the location for the next move.
   *  Postcondition: (1) The returned location is an element of locs, this critter's current location,
   *  or null. (2) The state of all actors is unchanged.
   *  @param locs the possible locations for the next move
   *  @return the location that was selected for the next move, or null to indicate
   *          that this OpossumCritter should be removed from the grid.
   */
  public Location selectMoveLocation(ArrayList<Location> locs)
  {  /* to be implemented in part (b) */  }
```

```
  /** @param other the actor to check
   *  @return true if other is a friend; false otherwise
   */
  private boolean isFriend(Actor other)
  {  /* implementation not shown */  }

  /** @param other the actor to check
   *  @return true if other is a foe; false otherwise
   */
  private boolean isFoe(Actor other)
  {  /* implementation not shown */  }
}
```

(a) Override the `processActors` method for the `OpossumCritter` class. This method should look at all elements of `actors` and determine whether or not to play dead according to the types of the actors. If there are more foes than friends, the `OpossumCritter` indicates that it is playing dead by changing its color to `Color.BLACK`. When not playing dead, it sets its color to `Color.ORANGE`. The instance variable `numStepsDead` should be updated to reflect the number of consecutive steps the `OpossumCritter` has played dead.

Complete method `processActors` below.

```
/** Whenever actors contains more foes than friends, this OpossumCritter plays dead.
 *  Postcondition: (1) The state of all actors in the grid other than this critter and the
 *  elements of actors is unchanged. (2) The location of this critter is unchanged.
 *  @param actors a group of actors to be processed
 */
public void processActors(ArrayList<Actor> actors)
```

(b) Override the `selectMoveLocation` method for the `OpossumCritter` class. When the `OpossumCritter` is not playing dead, it behaves like a `Critter`. The next location for an `OpossumCritter` that has been playing dead for three consecutive steps is `null`. Otherwise, an `OpossumCritter` that is playing dead remains in its current location.

Complete method `selectMoveLocation` below.

```
/** Selects the location for the next move.
 *  Postcondition: (1) The returned location is an element of locs, this critter's current location,
 *  or null. (2) The state of all actors is unchanged.
 *  @param locs the possible locations for the next move
 *  @return the location that was selected for the next move, or null to indicate
 *          that this OpossumCritter should be removed from the grid.
 */
public Location selectMoveLocation(ArrayList<Location> locs)
```