

## **PART A:**

```
public void processActors(ArrayList<Actor> actors)
{
    int numFriends = 0;
    int numFoes = 0;

    for (Actor nextActor : actors)
    {
        if (isFriend(nextActor))
            numFriends++;
        else if (isFoe(nextActor))
            numFoes++;
    }

    if (numFoes > numFriends)
    {
        setColor(Color.BLACK);
        numStepsDead++;
    }
    else
    {
        setColor(Color.ORANGE);
        numStepsDead = 0;
    }
}
```

## **PART B:**

```
public Location selectMoveLocation(ArrayList<Location> locs)
{
    if (numStepsDead == 3)
        return null;
    else if (numStepsDead > 0)
        return getLocation();
    else
        return super.selectMoveLocation(locs);
}
```

## **OR**

```
public Location selectMoveLocation(ArrayList<Location> locs)
{
    if (getColor().equals(Color.BLACK))
    {
        if (numStepsDead == 3)
            return null;
        else
            return getLocation();
    }
    return super.selectMoveLocation(locs);
}
```

<b>Part A:</b>	<code>processActors</code>	<b>6 points</b>
----------------	----------------------------	-----------------

- +1/2 initialize friend/foe counter(s)
- +2 1/2 loop and identify actors
  - +1 traverse `actors`
    - +1/2 correctly access an element of `actors` (in context of loop)
    - +1/2 access all elements of `actors` (lose this if index out-of-bounds)
  - +1 1/2 identify actor category and update counters (in context of loop)
    - +1/2 call `isFriend(nextActorFromList)`
    - +1/2 call `isFoe(nextActorFromList)`
    - +1/2 update counters appropriately in both cases
- +3 update `OpossumCritter` state
  - +1 correctly identify whether to play dead
  - +1 appropriate result if playing dead
    - +1/2 `setColor(Color.BLACK)`
    - +1/2 `numStepsDead++`
  - +1 appropriate result if normal
    - +1/2 `setColor(Color.ORANGE)`
    - +1/2 `numStepsDead = 0`

<b>Part B:</b>	<code>selectMoveLocation</code>	<b>3 points</b>
----------------	---------------------------------	-----------------

- +1 determine appropriate case (using `==` with `Color` is okay)
  - +1/2 correctly identify one case (dead, playing dead, normal)
  - +1/2 correctly identify all three cases
- +2 appropriate return values
  - +1/2 return null if really dead
  - +1/2 return current location if playing dead
  - +1 return `super.selectMoveLocation(locs)` otherwise
    - +1/2 `super.selectMoveLocation(locs)`
    - +1/2 return value from call

Usage:

- 1 if violate postconditions (e.g., `removeSelfFromGrid()`)
- 1 for `BLACK` or "Black" instead of `Color.BLACK`
- 1/2 for call to (nonexistent) default `Location` constructor