Use the [GridWorld Quick Reference](https://secure-media.collegeboard.org/apc/ap_comp_sci_a_quick_reference.pdf) if needed.Test the code out during and after each step to make sure it is error free

**Overview:**

A LeapFrogActor will move forward until it reaches another Actor, then it will jump over the other Actor. When it jumps over the other Actor it removes that Actor from the world. A LeapFrogActor cannot jump out of the world and it can only jump over one Actor. If the actor reaches the edge or cannot jump over then it will turn either direction and keep moving forward.

**Part 0:**

Create a new Class named LeapFrogActorWorld w/ a main method

Create a new ActorWorld.

As you go through this lab, add actors, rocks, flowers, bugs, critters, to the world as needed to test if the LeapFrogActor is working.

New Classes/Methods to use:

new ActorWorld( ) or new ActorWorld( new BoundedGrid<Actor>( int r, int c))

world.show()

**\*\* Test part 0 out before moving on to part 1 \*\***

**Part 1:**

Create a new Class named LeapFrogActor, a LeapFrogActor is an Actor.

Write a default constructor for the LeapFrogActor that sets it to green.

Go to the LeapFrogActorWorld and add a default LeapFrogActor to the location (0,0)

New Classes/Methods to use:

setColor( Color color )

world.add( Location loc, Actor actor )

new Location( int row, int col )

new LeapFrogActor( )

**\*\* Test Part 1 out before moving on to part 2 \*\***

**Part 2:**

Override the act method to make the LeapFrogActor move forward. (Do **NOT** write any other methods!!!!)

Check if the next adjacent location that it is facing is valid. If it is then move to it, otherwise change direction.

New Classes/Methods to use:

\*\*Hint: look at the Bug class’s method for a reference\*\*

getLocation(), getGrid(), getDirection()

*Location* >> getAdjacentLocation( Direction dir )

*Grid* >> isValid( Location loc )

moveTo( Location newLocation )

setDirection( Direction dir )

Location >> All directional static final variables

**\*\* Test Part 2 out before moving on to part 3 \*\***

**Part 3:**

Make the LeapFrogActor “jump” over another Actor.

Add an Actor to the world at (0,3).

Check if the next location has an Actor in it. If so, jump it and remove it from the world!

Also, check if the location that you are jumping to is a valid location to move. If it is not then you need to turn.

***Hint:*** Make variables to hold the locations and actors that you are checking.

New Classes/Methods to use:

world.add( Location loc, Actor a ), new Location( int r,int c ), new Actor()

Grid >> get( Location loc ) /\*this returns the Actor in that Location, if no Actor it returns null\*/

*[whatever]* instanceof Actor /\*this returns true if whatever is an instance of an Actor \*/

**\*\* Test part 3 out before moving on to part 4 \*\***

**Part 4:**

A leapFrogActor can only jump over one actor at a time. So if there are multiple actors in the way then make it turn. Before you jump an actor, you need to check if the location you are jumping to contains an actor. If it does then it needs to turn.

***Hint:*** Use the location of the Actor you are wanting to jump and the direction of the leapfrog actor.

New Classes/Methods to use:

/\*no new methods or classes that have not been used\*/

**\*\* Test part 4 out before moving on to part 5 \*\***

**Part 5:**

Add the Actor that is jumped over and gets removed to a list of Actors.

***Hint:*** Make an instance variable.

New Classes/Methods to use:

ArrayList<Actor>

**\*\* Test part 5 out before turning it in \*\***