Step3

1.How would you access the row value for loc1?

Loc1.getRow();

2.What is the value of b after the following statement is executed?

```
boolean b = loc1.equals(loc2);
```

B = true.

3.What is the value of loc3 after the following statement is executed?

```
Location loc3 = loc2.getAdjacentLocation(Location.SOUTH);
```

Loc3 = (4, 4).

4.What is the value of dir after the following statement is executed?

```
int dir = loc1.getDirectionToward(new Location(6, 5));
```

Dir = 135.

5.How does the getAdjacentLocation method know which adjacent location to return?

There is a parameter to getAdjacentLocation method, and this parameter indicates which adjacent location should be returned.

Step4

1.How can you obtain a count of the objects in a grid? How can you obtain a count of the empty locations in a bounded grid?

```
ArrayList<Location> arr  = new ArrayList<Location>();
arr = grid.getOccupiedLocations();
int count = arr.length();
```

2.How can you check if location (10,10) is in a grid?

```
Boolean b = grid.isValid(new Location(10, 10));
```

3.Grid contains method declarations, but no code is supplied in the method. Why? Where can you find the implementations of these methods?

We can find the implementations of these methods in the classes

that implement Grid interface.

4.All method that return multiple objects return them in an ArrayList. Do you think it would a better design to return the objects in an array? Explain your answer.

I don't think it's better design to return the objects in array, because once the array is declared, its length is determined and can't be changed any more; but we don't know the number of objects will be returned.

Step5

1.Name three properties of every actor?

Location, Direction, Color.

2.When an actor is constructed, what is its direction and color?

Direction:North Color:Blue

3.Why do you think that the Actor class was created as a class instead of an interface?

Because for classes which extend Actor class, they have some common methods and just need to override part of methods of Actor, and remains the rest. But if Actor is an interface, then we need to implements all of its method in every class implements Actor, which will be too troublesome.

4.Can an actor put itself into a grid twice without first removing itself? Can an actor remove itself from a grid twice? Can an actor be placed into a grid, remove itself, and then put itself back? Try it out. What happens?

An actor can't put itself into a grid twice, which will cause an IllegalStateException.

An actor can't remove iteself from a grid twice, because once it is removed, we can't find the actor in the grid, then we can't remove it from the grid again.

An actor can't remove iteself from a grid, and then put itself back, because once it is removed, we can't find it in the grid and then can't put it back.

5.How can an actor turn 90 degree to the right?

If an actor wants to turn 90 degree, it has to call the turn()

method twice.

Step6

1. Which statement(s) in the canMove method ensures that a bug does not try to move out of its grid?

If (!gr.isValid(next)) return false;

2.Which statement(s) in the canMove method determines that a bug will not walk into a rock?

Return (neighbor == null) || (neighbor instanceof Flower);

3.Which methods of the Grid interface are invoked by the canMove method and why?

Boolean is Valid(Location loc);

Because we need to check whether the new location that a bug is going to move to is in the grid.

E get(Location loc);

We use these this method to get the neighbors of a bug that is moving.

4.Which method of the Location class is invoked by the canMove method and why?

Public Location getAdjacentLocation(int direction); We can use this method to get the Location in front of the bug that is moving.

5. Which methods inherited from the Actor class are invoked in the canMove method?

Public Grid<Actor> getGrid( );

public Location getLocation( );

6.What happens in the move method when the location immediately in front of the bug is out of the grid?

It will call removeSelfFromGrid method to remove the actor from the grid.

7.Is the variable loc needed in the move method, or could it be avoided by calling getLocation() multiple times?

We need the variable loc need in the move method, such that we only need to call the getLocaiton method for one time, which will make our program more effcient.

8.Why do you think the flowers that are dropped by a bug have the same color as the bug?

Because from the move method of a bug, we can new that the flower is newed with the color of the bug;

```
Flower flower = new Flower(getColor());
```

9.When a bug removes itself from the grid, will it place a flower into its previous location?

Yes.

10. Which statement(s) in the move method places the flower into the grid at the bug's previous location?

```
Flower flower = new Flower(getColor());
flower.putSelfInGrid(gr, loc);
```

11.If a bug needs to turn 180 degrees, how many times should it call the turn method?

Four times.