APCS :: Lab 8.2

SuperBot Additional Functionality

All of the methods you are being asked to create are fairly easy to write using recursion. Try to come up with a simple recursive way to describe the problem solution before you begin!

Now that you know how to use arguments to a method, add some functionality to SuperBot:

- **Overload the move() method** so that if there is an integer argument, the robot will move that many blocks. Your SuperBot should already be robust enough not to run into any walls that might be in the way.

- **Overload the putBeeper() method** so that if there is an integer argument, the robot will put down that many beepers. Your SuperBot should already be robust enough not to try to put down a beeper if its bag is empty.

- **Overload the pickBeeper() method** so that if there is an integer argument, the robot will pick up that many beepers. Your SuperBot should already be robust enough not to try to pick up a beeper that’s not there.

Now that you know how to write methods that return values, add more functionality to SuperBot with the following methods:

- **numBeeepersInBag()**
  Returns the number of beepers the robot has in its bag. At the end, the robot should have the same number of beepers it started with.

- **numBeeepersOnCorner()**
  Returns the number of beepers on the corner (intersection) the robot is standing on. At the end, the corner should have the same number of beepers it started with.

- **distanceToWall()**
  Returns the number of blocks to the first wall in the robot’s path. You can assume that there is a wall in the way. At the end, the robot should be on the same corner it started on.

- **distanceToBeeper()**
  Returns the number of blocks to the first beeper in the robot’s path. You can assume that there is a beeper in the way. At the end, the robot should be on the same corner it started on.