Start out with the SuperBot we started creating in class (formerly known as BeepKeeper).

Part 0: recursion practice
Write a method called move(int numSteps), which recursively moves the number of steps given in the parameter ‘numSteps.’ The recursive definition of move(int numSteps) is as follows:
1. If numSteps == 0 return;
2. otherwise,
   do a standard move
   move(numSteps-1)

Once you’ve done that, then modify the move(int numSteps) method so that if a negative number is given, the robot moves backward that many steps. You can think of the recursive method this way:
If numSteps == 0
then stop
Else if numSteps > 0
then move() and recursively call numSteps
Else
Turn around,
recursively call move with numSteps having the opposite sign
turn around again.

Part 1:
1.1 Add a recursive method called void putBeepers(int numToPut). The method’s argument is the number of beepers that the robot should put down. Prevent the program from crashing by checking to see if there are anyBeepersInBeeperBag before putting a beeper down.

1.2 Add a private member variable that keeps track of the number of beepers the robot has in its bag. (Remember to account for the beepers the robot starts out with). When a beeper is put down the number should decrement, when a beeper is picked up it should increment.

1.3 Add a method called int getNumBeeper() which returns the number of beepers (yes, this will be a one line method).

1.4 Modify your putBeepers method so that the robot won’t put down any beepers unless it has enough to do so. For example, if the robot has 8 beepers, and a command is issued to putBeepers(10) the robot should not put any beepers down. Instead, print an error message to the console.

Part 2:
2.1 Add two member variables to keep track of the robot’s street and avenue. Modify the move method so that it updates these variables to accurately reflect the robot’s current location. (NOTE:
if the robot moves and it’s facing north immediately after moving, then its street should increment by one, and its avenue would stay the same.

2.2 Add two methods: int getStreet() and int getAvenue() that return the robot’s current street and avenue respectively.

2.3 Write a method called void moveTo(int st, int ave) which attempts to move the robot from it’s current location to the location given in the method parameters. HINT: Always make the robot face a particular direction before moving it – it makes the math easier. It IS NOT required that the robot be facing any particular direction once at the location, only that the robot is at the location.