vanand and fsikram

*****SUMMARY*****

The program basically implements a 2D predator-prey simulation using derived classes and virtual functions. The program involves a simulation of a grid of n by n squares, which could be empty or occupied by an organism. All the organisms are placed in a vector in Gameboard.The two kinds of organisms are ants and doodlebugs. Only one organism takes one cell at a time and each organism performs some action for every step. The organism may never move off the edge. The actions the doodlebug performs is starving, moving, eating, and breeding. The actions the ant does is moving and breeding. The Organism class is an abstract class. The ant and doodlebug is a subclass of the Organism class. The Organism class has common virtual methods between the ant and doodlebug class that they can implement. We have a header and .cpp file for ant, doodlebug, organism, gameboard, and the main program.

Note: The move function that is called on board while we are traversing through timesteps does ALL of the following: breed for ant and doodlebug, move ant and doodlebug, and starve doodlebug.

*****HOW TO RUN IT*****

In order to run the program, first you have to compile it. In the terminal, you can type make after going to the folder. To actually run the program, you have to type in the following command line arguments. These are the file name, the grid size, the number of doodlebugs, the number of ants, the time, the seed number, and pause. You can also not enter the command line arguments because they all have a default value. The run line will look like this in the terminal:

./PA6 gridSize #doodlebugs #ants #time_steps seed pause

Note: When you pause you MUST type a letter and press enter to continue the program

*****PROBLEMS*****

We did have a few problems initially when we wanted to start. We didn't know where to start but then we read the book and talked to the TA's and professor and we got an idea of where to start. After we started, we had a problem with the header files as they were looping back to each other after running the program. We went to the TA's to discover that there is something called forward declaration. We were then able to make that part work.

******ALGORITHM******

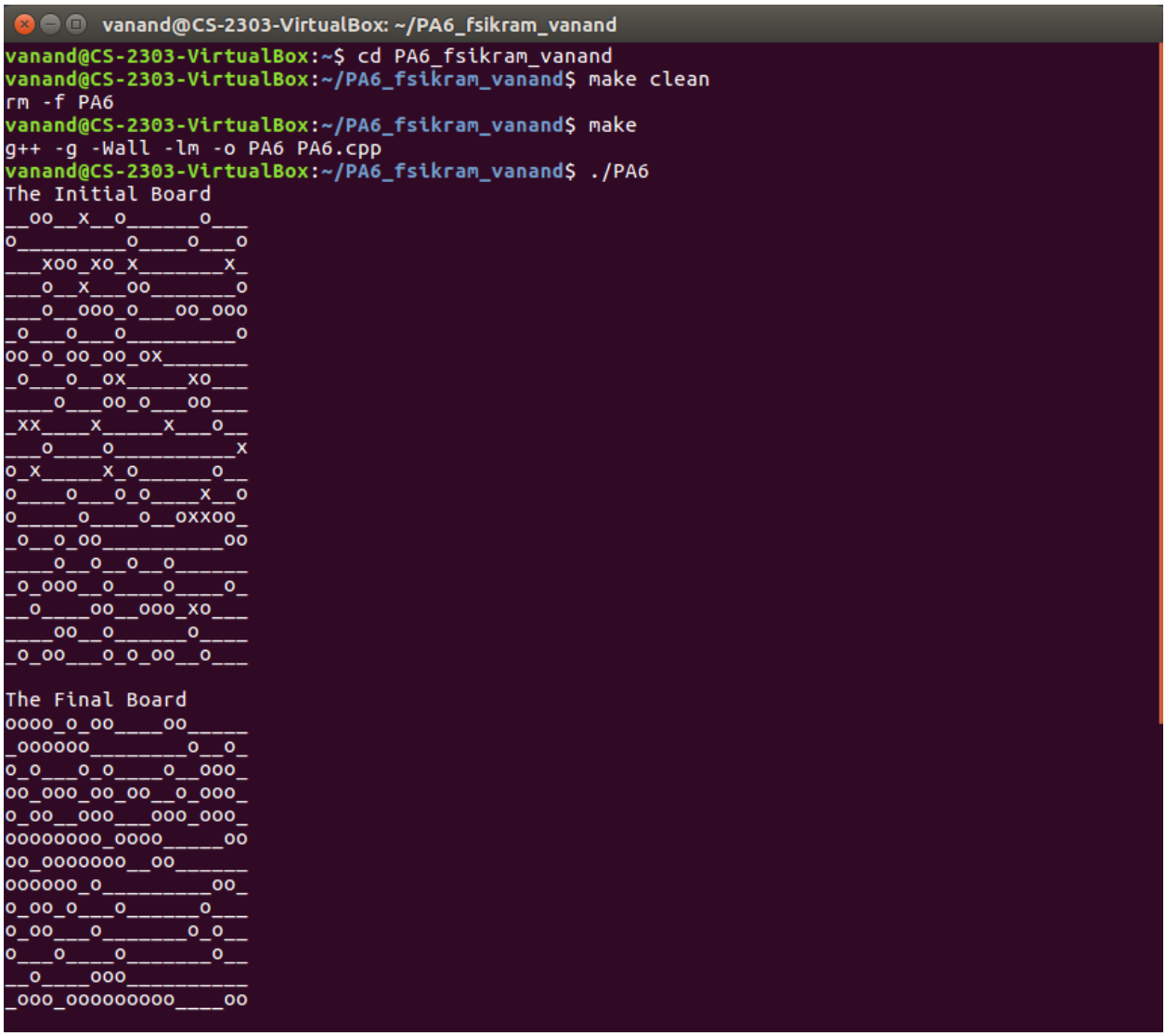
As part of our algorithm we have the interpretation of the command line arguments and the initial configuration of the board. We have a function that finds the random number. We have a function for adding ants and doodlebugs to the board and it randomly places them on the board. It creates new ants and doodlebugs. We have a min function that basically plays the simulation by calling the methods such as breed, move, starve. For move, set the direction equal to 1 for moving above, 2 for moving down, 3 for moving left and 4 for moving right. Calls the set organism and get organism function to set the organism in the board and get the organisms, respectively. For breed, create a vector to hold the spots available on board. Randomize the spot it will breed in if many adjacent cells are open. Then call helper function to determine exactly which spot. Then, if statements that tell where to put the breed and set that in place.

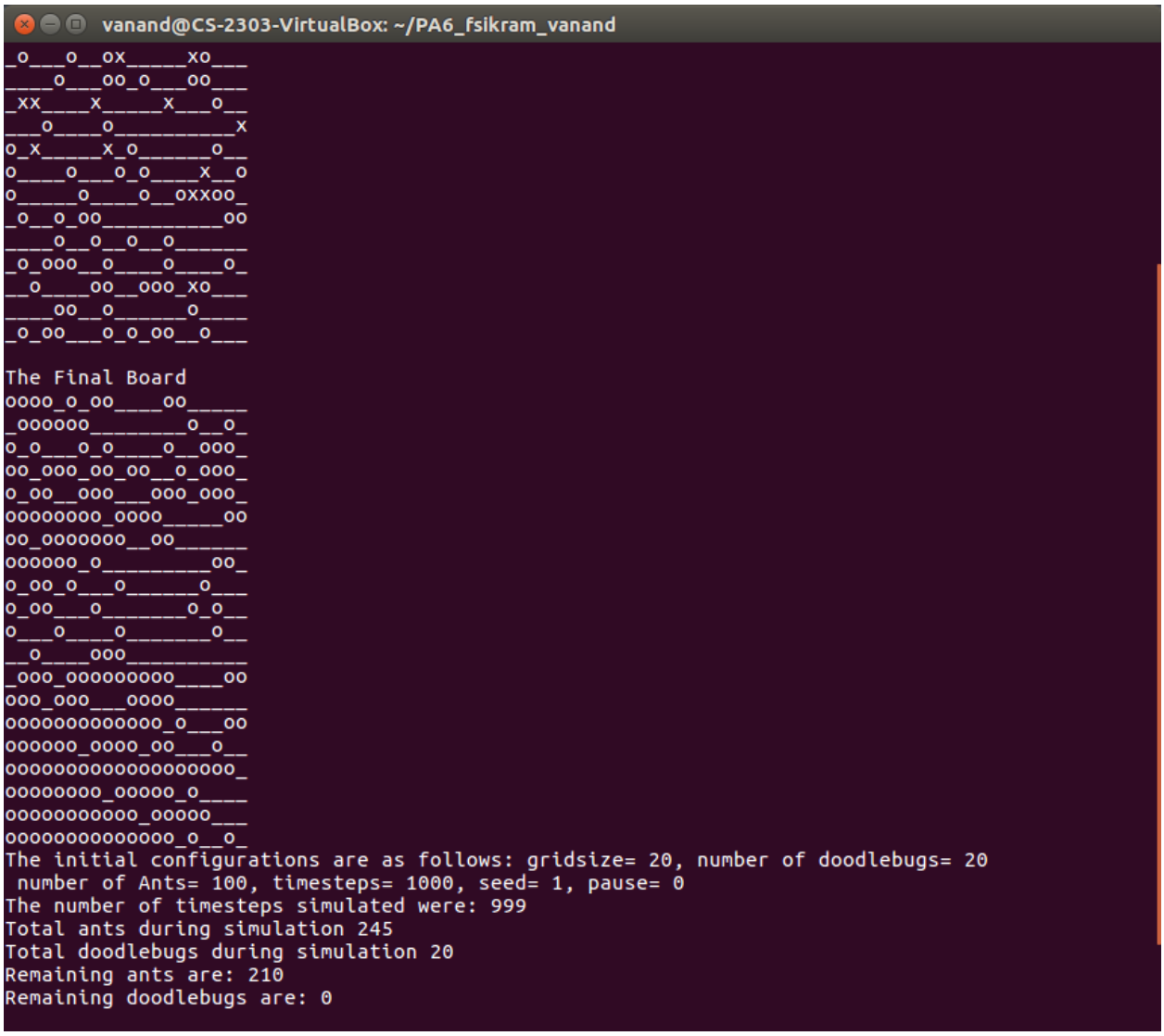
Then, reset the lifespan of the organism to start over again. Then, if statement to differentiate between an ant and a doodlebug.

*****TESTING*****
We tested all the cases that the program could encounter. The command line arguments are displayed below for each test:

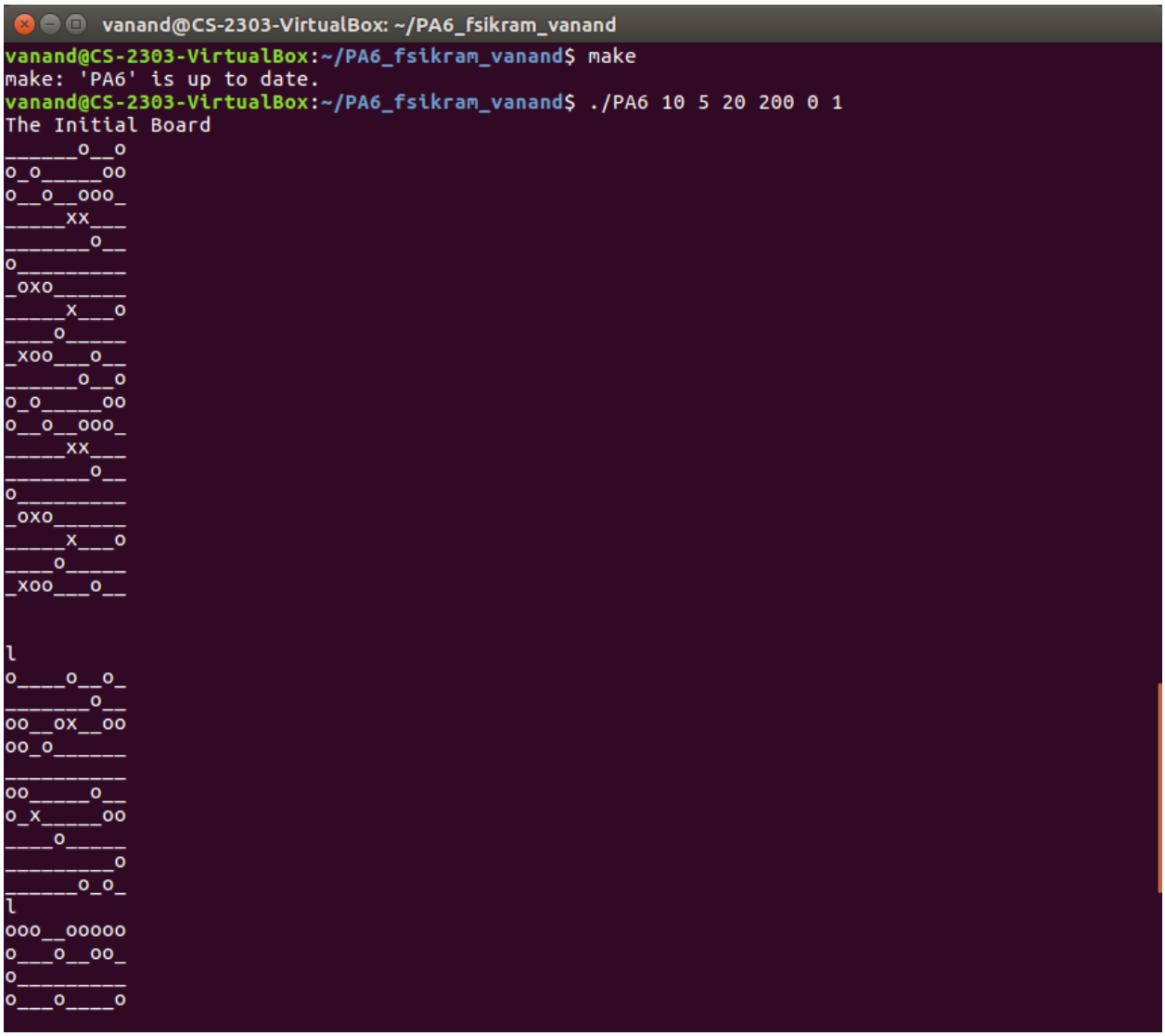
*****OUTPUT*****

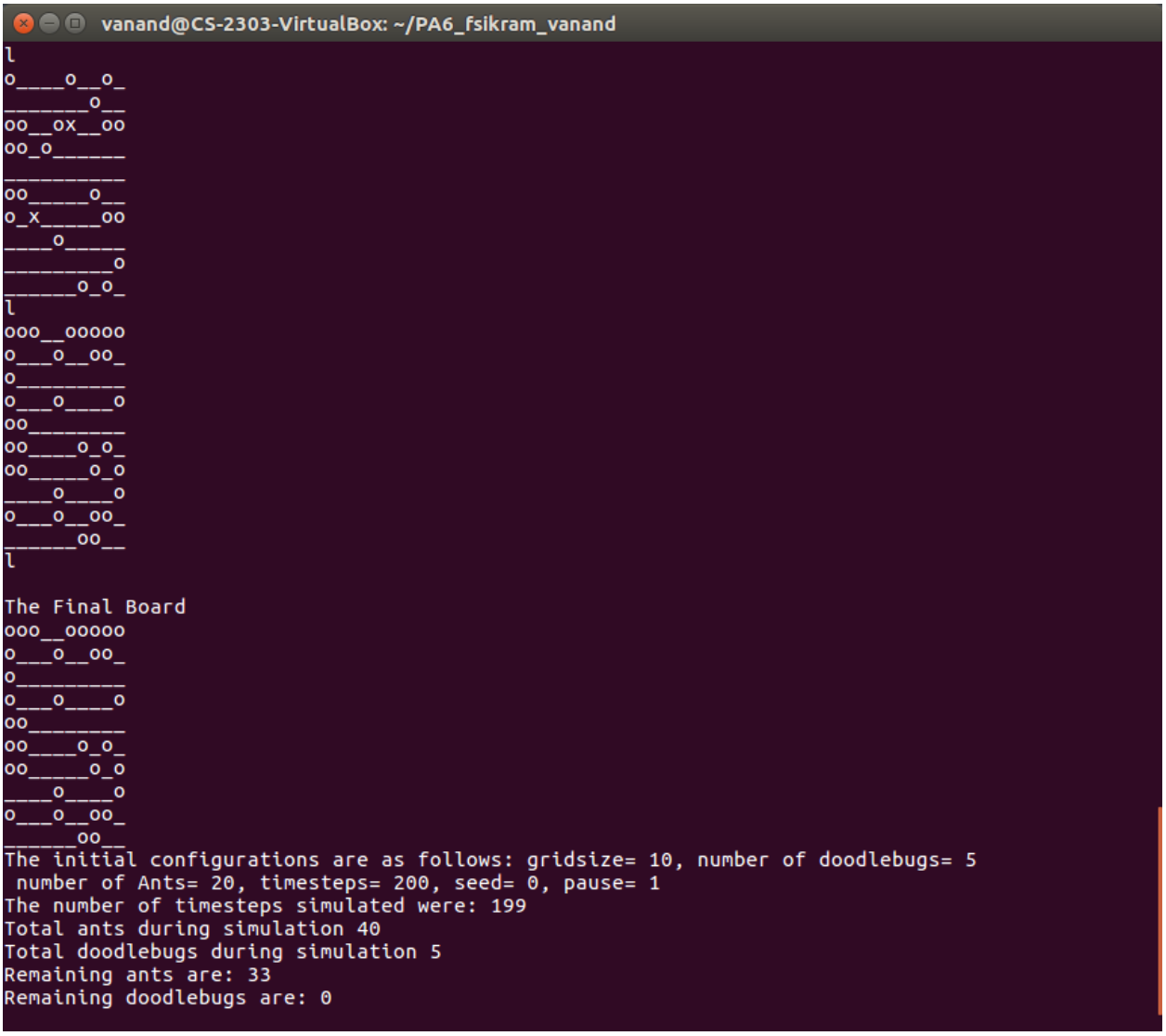
Test case 1: Default values



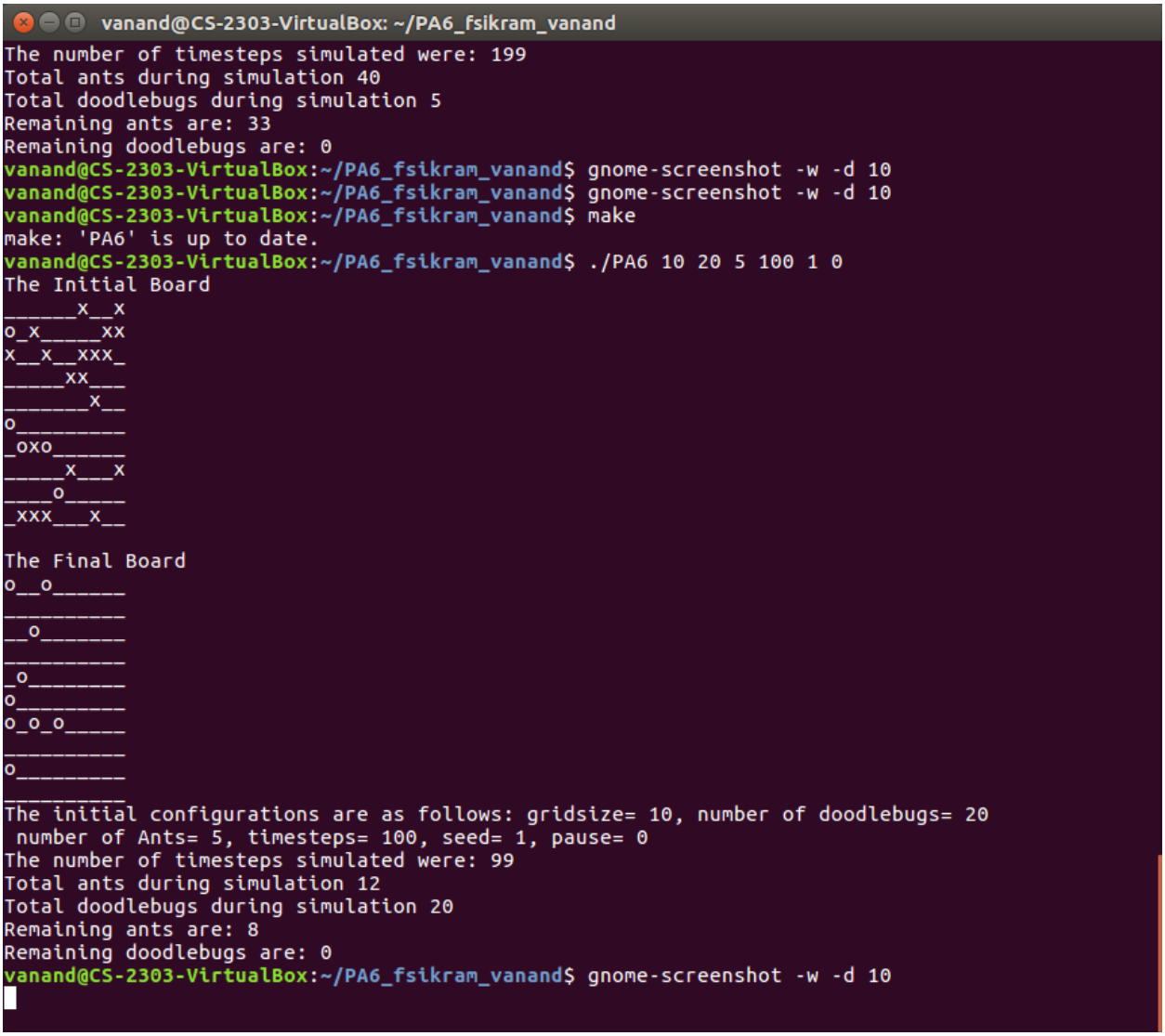

vanand@CS-2303-VirtualBox: ~/PA6_fsikram_vanand
_o___o__ox_____xo___
____o___oo_o___oo___
_xx____x_____x___o__
___o____o__________x
o_x_____x_o______o__
o____o___o_o____x__o
o_____o____o__oxxoo_
_o__o_oo__________oo
____o__o__o__o______
_o_ooo__o____o____o_
__o____oo__ooo_xo___
____oo__o______o____
_o_oo___o_o_oo__o___
The Final Board
oooo_o_oo____oo_____
_oooooo________o__o_
o_o___o_o____o__ooo_
oo_ooo_oo_oo__o_ooo_
o_oo__ooo___ooo_ooo_
oooooooo_oooo_____oo
oo_ooooooo__oo______
oooooo_o_________oo_
o_oo_o___o______o___
o_oo___o_______o_o__
o___o____o_______o__
__o____ooo__________
_ooo_oooooooooo____oo
ooo_ooo___oooo______
ooooooooooooo_o___oo
oooooo_oooo_oo___o__
ooooooooooooooooooo_
oooooooo_ooooo_o____
ooooooooooo_ooooo___
ooooooooooooooo_o__o_
The initial configurations are as follows: gridsize= 20, number of doodlebugs= 20
 number of Ants= 100, timesteps= 1000, seed= 1, pause= 0
The number of timesteps simulated were: 999
Total ants during simulation 245
Total doodlebugs during simulation 20
Remaining ants are: 210
Remaining doodlebugs are: 0

Test Case 2: Doodlebug is less than Ant

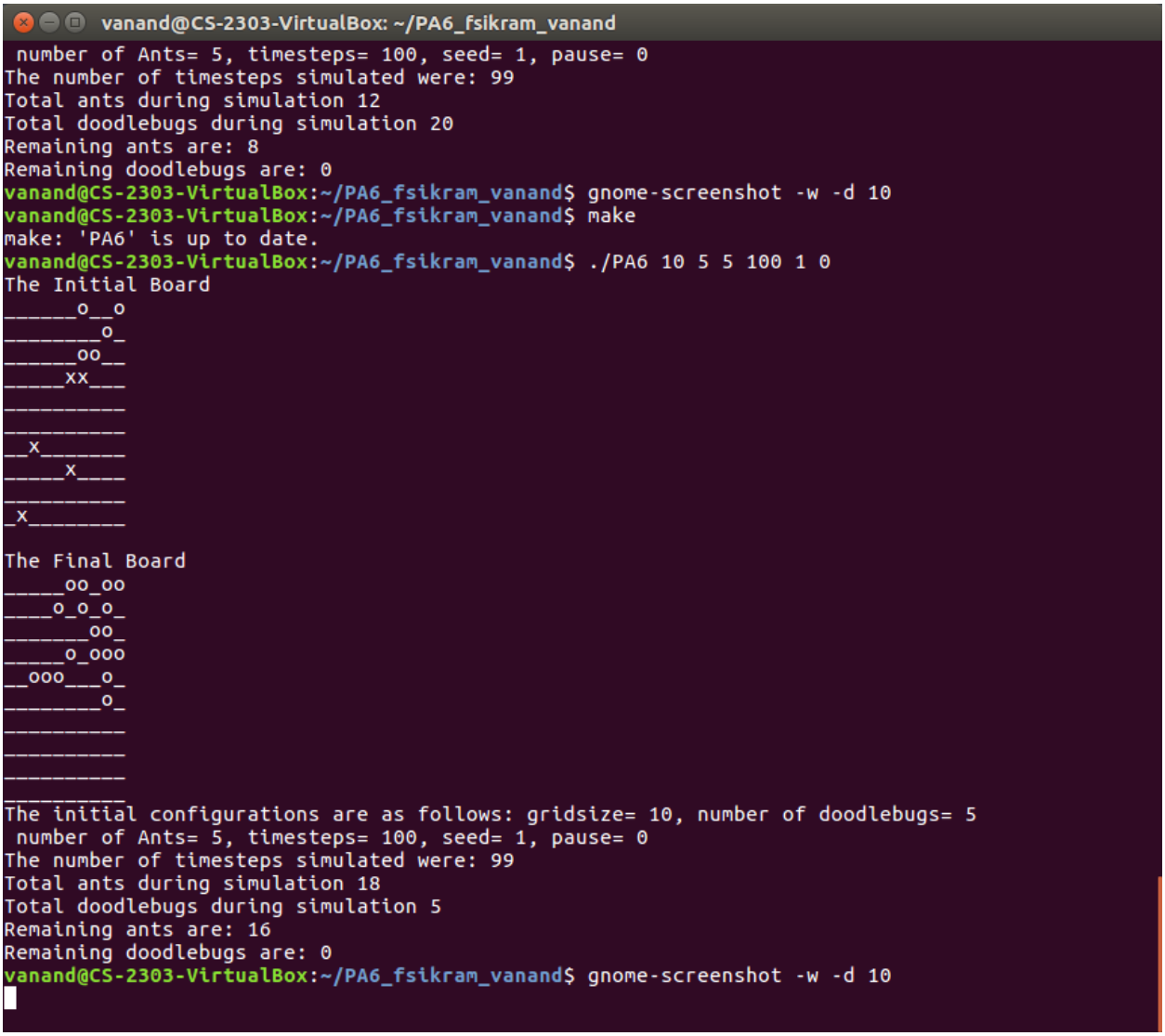



vanand@CS-2303-VirtualBox: ~/PA6_fsikram_vanand
l
o____o__o_
_______o__
oo__ox__oo
oo_o______
____
oo_____o__
o_x_____oo
____o_____
_________o
______o_o_
l
ooo__ooooo
o___o__oo_
o_________
o___o____o
oo________
oo____o_o_
oo_____o_o
____o____o
o___o__oo_
______oo__
l
The Final Board
ooo__ooooo
o___o__oo_
o_________
o___o____o
oo________
oo____o_o_
oo_____o_o
____o____o
o___o__oo_
______oo__
The initial configurations are as follows: gridsize= 10, number of doodlebugs= 5
number of Ants= 20, timesteps= 200, seed= 0, pause= 1
The number of timesteps simulated were: 199
Total ants during simulation 40
Total doodlebugs during simulation 5
Remaining ants are: 33
Remaining doodlebugs are: 0

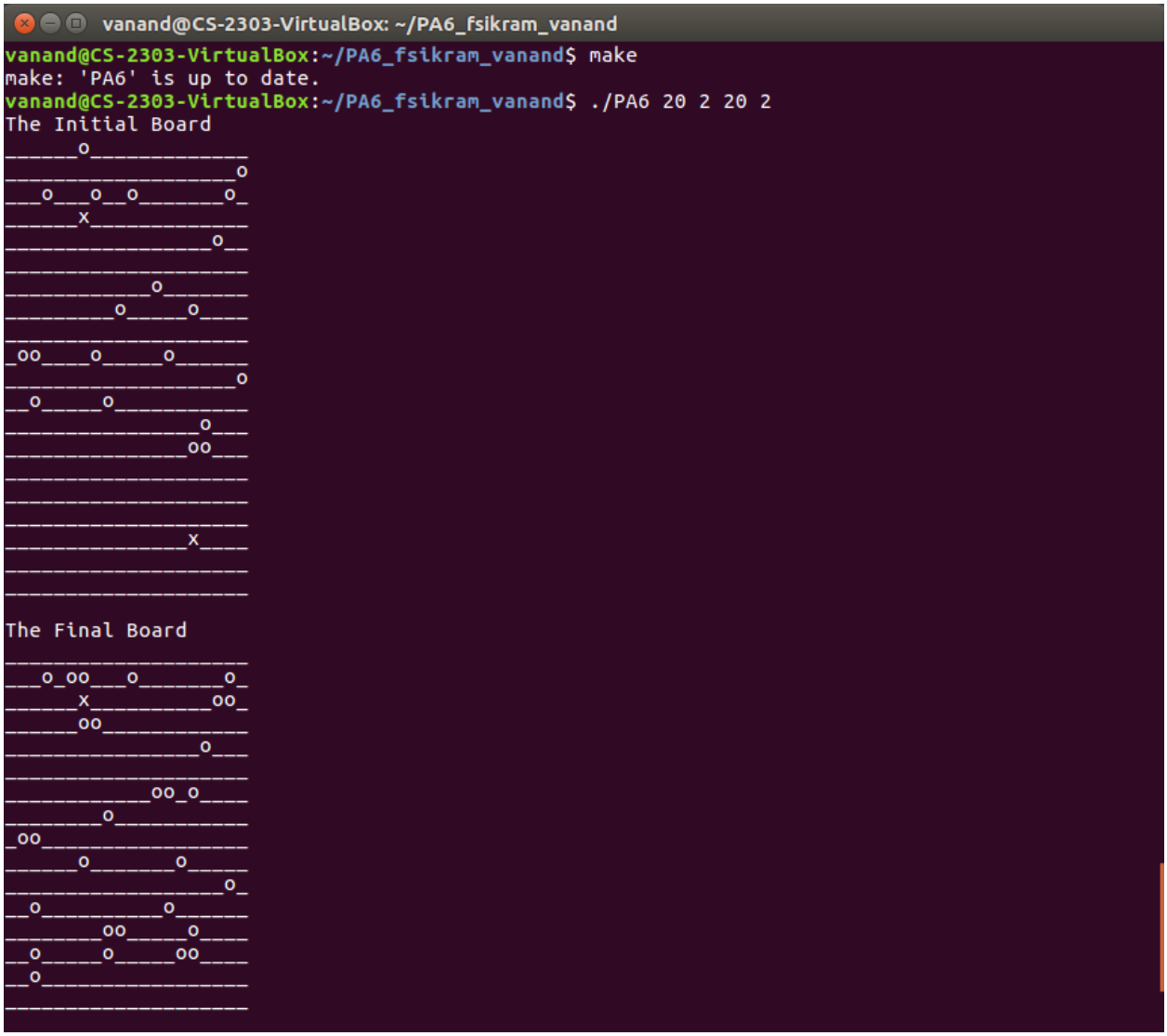
Test Case 3: Doodlebug is more than Ant

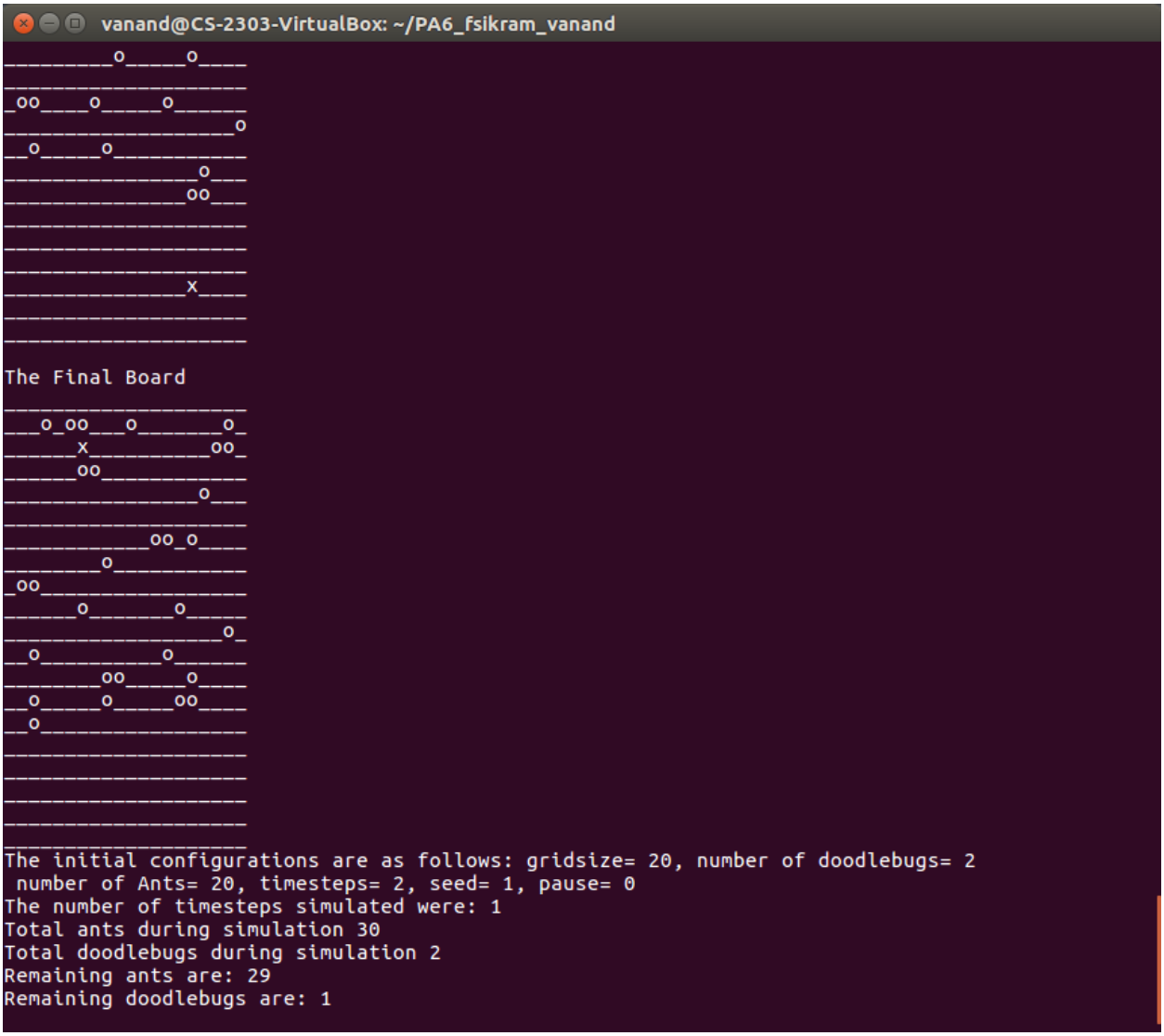


Test Case 4: Same amount of doodlebug and ant



Test Case 5: One step




vanand@CS-2303-VirtualBox: ~/PA6_fsikram_vanand
_____o_____o____
_____
_oo____o_____o_____
_____o
__o_____o_____
_____o___
_____oo___
_____
_____
_____
_____x____
_____
_____
The Final Board
_____
___o_oo___o_____o_
_____x_____oo_
_____oo_____
_____o___
_____
_____oo_o____
_____o_____
_oo_____
_____o_____o_____
_____o_
__o_____o_____
_____oo_____o____
__o_____o_____oo____
__o_____
_____
_____
_____
_____
The initial configurations are as follows: gridsize= 20, number of doodlebugs= 2
number of Ants= 20, timesteps= 2, seed= 1, pause= 0
The number of timesteps simulated were: 1
Total ants during simulation 30
Total doodlebugs during simulation 2
Remaining ants are: 29
Remaining doodlebugs are: 1