Homework 13: Doodlebug

CH16# 9:

The goal for this Project is to create a simple two-dimensional predator-prey simulation. In this simulation the prey are ants and the predators are doodlebugs. These critters live in a world composed of a 20 × 20 grid of cells. Only one critter may occupy a cell at a time. The grid is enclosed, so a critter is not allowed to move off the edges of the world. Time is simulated in time steps. Each critter performs some action every time step.

The ants behave according to the following model:

<u>Move</u>: Every time step, randomly try to move up, down, left, or right. If the neighboring cell in the selected direction is occupied or would move the ant off the grid, then the ant stays in the current cell.
<u>Breed</u>: If an ant survives for three time steps, then at the end of the time step (that is; after moving) the ant will breed. This is simulated by creating a new ant in an adjacent (up, down, left, or right) cell that is empty. If there is no empty cell available, then no breeding occurs. Once an off-spring is produced, an ant cannot produce an offspring until three more time steps have elapsed.

The doodlebugs behave according to the following model:

<u>Move</u>: Every time step, if there is an adjacent ant (up, down, left, or right), then the doodlebug will move to that cell and eat the ant. Otherwise, the doodlebug moves according to the same rules as the ant. Note that a doodlebug cannot eat other doodlebugs.
<u>Breed</u>: If a doodlebug survives for eight time steps, then at the end of the time step it will spawn off a new doodlebug in the same manner as the ant.
<u>Starve</u>: If a doodlebug has not eaten an ant within the last three time steps, then at the end of the third time step it will starve and die. The doodlebug should then be removed from the grid of cells.

During one turn, all the doodlebugs should move before the ants do.

Write a program to implement this simulation and draw the world using ASCII characters of "o" for an ant and "X" for a doodlebug or "-" for an empty space. Create a class named Organism that encapsulates basic data common to both ants and doodlebugs. This class should have a virtual function named move that is defined in the derived classes of Ant and Doodlebug. You may need additional data structures to keep track of which critters have moved.

Your task: Initialize the world with 5 doodlebugs and 100 ants. After each time step, prompt the user to press Enter to move to the next time step. You should see a cyclical pattern between the population of predators and prey, although random perturbations may lead to the elimination of one or both species. Please submit a single .cpp; do NOT separate this into multiple .h files. The .cpp should contain all classes and structures needed so that it can execute when running the main function.

An example of a program timestep output is given below. It does not need to look exactly like this but this can be a good starting point if you are confused.

```
World at time 0:

o - o o o o o - - o - - - - - - o o - o o
- - - - o - - o X - - - - o - - o - - -
- - - - - - - o o - o - - - o - o - o -
- - - - - o - - o - - - - - - - - - - -
- o - - - - o - - - o - - - - - - o -
- - - - - - - o o - - o o o o - - - - -
- - - o o - - - o - - - - o - - - - -
- - - - - - - X - o - o - - - - o - o -
- - - - - - - o - - o - o o - - - - - -
- - - o - - - - - - - o o - - - - - - X
- - - - o - - - o o - o - o - - - - - -
- - - - - - - - - o - - - - o - - - - -
o - - - - - - - o - - - - - - - - - - -
- o - o - - - o - - o - X - - - - - - -
o o - o - o - o - - o - - o - o - o o o
- - - o - - o - - o - o - o - - o - - o
o - - - - - - - - - o - - o - - - - - o
o - - o - - o o - o o - - - - - - - - o
- - o - - - X o - - - - - - - o - - o o o
- o - - o o o - - - - - - - - - - - - -

Press ENTER to continue
```