Programming Assignment #6

Due:12/11/2022 23:59:59

Topic: Design a simulator for an artificial life game with goats and grass.

Objective: Practice polymorphism in class inheritance.

Description:

In this assignment, you will design a simulation of predatory and prey relationship where goats are predators and grass is the prey. Goats appear in the console window of fixed size as X's and grass as I's. (Examples are shown in the following section.) The "world" consists of a 35x20 array. All creatures live in cells of this array. The world does not wrap around, i.e., there are firm boundaries at the edges of the array. Initially there are 5 goats and 10 blades of grass scattered randomly. As time processes, each creature in the world takes its turn to act for living. The complete set of turns by all living creatures is called a *pass*. During each pass, goats move around and eat grass while the grass grows.

During a pass, each goat is allowed to move in a randomly selected direction, provided the square is not currently occupied by another goat or outside the boundaries of the world. When the attempt is not *legal*, the goat does not get another chance to try again. That is, a goat has only one chance to move in each pass. When a goat moves to a new cell, it gets to eat any grass on the cell. Each goat starts with 20 food points and consumes 1 point with each pass. Every time the goat eats a blade of grass it gains 5 points.

Goats also age (become older). When a goat is born, it is zero days old. It then grows one day older with each pass until it dies (disappear) at the age of 70 days. However, between the ages of 50 and 55 inclusively, a goat can have a baby goat for each pass. To do so, the mommy goat picks a random direction and generates a baby goat one cell away in that direction, provided that the cell is legal. If the square contains a blade of grass, the mommy goat also gets to eat the grass and acquires the usual food points. However, the mommy goat does not move during this process. There are no daddy goats and we won't worry about how the goats get pregnant. Goats die if their food points reach zero or their lives are over 70 days.

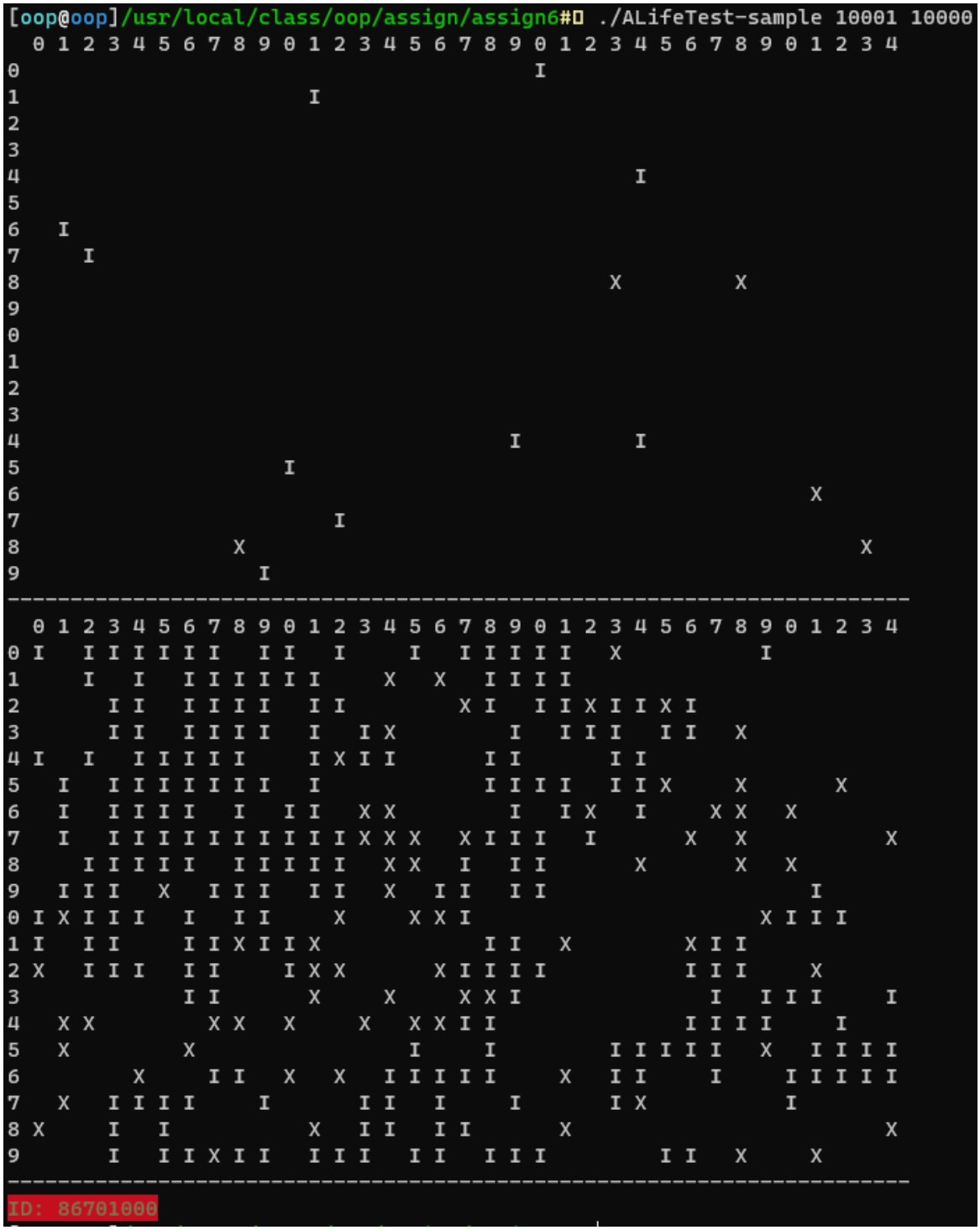
The behavior of grass is simpler. Grass never moves and doesn't eat. Grass starts out with an age of zero and grows one day older with each pass. When a blade of grass reaches 6 days, it dies (disappears). Grass can sprout new grass between the age of 3 and 5 inclusively. Like goats, a mommy grass picks a random direction and generates a baby grass at that neighboring cell if it is empty. If it fails, it does not do another attempt.

During each pass of the simulation, the program scan the world row by row from left to right and call upon each creature found to act. You have to make sure that each creature only acts once in each pass even if it has moved. For example, a goat may move to a new cell that will be scanned in a later time. In this case, you must have a way of jumping over that goat. In addition, baby goats and baby grass do not act during their first day of existence; they are merely by-products of existing creatures.

A sample interface is shown as below. I suggest you to play with the sample executable first to get a feel of what it does. The user can specify three optional parameters from the command line. The first parameter is the number of passes to run. The second parameter is the number of passes between every display of the world. The third parameter is the seed for the random number generator. For example, `AlifeTest-sample 1000 100 1234` means that the simulation will run for 1000 passes and display the world once every 100 passes with the seed of 1234 for the random number generator. If the user specifies zero passes, or a negative number, the program defaults to a single pass.

Assignment Directory: `/usr/local/class/oop/assign/assign6`

Sample Output:



Implementation Notes:

The design of this program is largely up to you; however, the random number generator class (`RandomNum.h` and `RandomNum.cc`) and the main testing function (`ALifeTest.cc`) is given to you. You are encouraged to design your class member carefully such that a member tries to be as `private`, `protected`, and then `public` in this order. We expect all derivations to follow the **IS-A** rule. In addition, remember that the purpose of this assignment is to practice polymorphism (e.g. virtual function) in class inheritance. Therefore, how good you have designed this part of class will be an important criterion in grading your program.

What to Hand in:

As usually, you need to submit a complete program electronically.