Project Assignment
Ecosystem Simulation

General description

This project involves creating a simulation of a simple ecosystem containing foxes, rabbits, and rocks. The simulation demonstrates predator-prey dynamics and environmental interactions in a controlled virtual environment.

The simulation should be done using a matrix of R rows and C columns (the ecosystem world), where each cell represents the ecosystem space on which foxes and rabbits move. In each generation, each cell may be free or occupied. A cell can be occupied by a fox, a rabbit or a rock. The world is initialized with a given population of foxes, rabbits and rocks which, over the generations, evolves according to a fixed set of rules. The simulation starts with a given initial ecosystem, and builds the successive generations of foxes and rabbits until a given number of generations (N_GEN) is reached. The problem to be solved consists in the simulation of a simple ecosystem inhabited by two species of animals, foxes and rabbits, whose existence over generations is mutually dependent.

Components

- Rabbits: Herbivores that reproduce quickly and serve as food for foxes
- Foxes: Carnivores that hunt rabbits to survive and reproduce
- Rocks: Static environmental elements that provide shelter and affect movement

Rules for Rabbits

- Rabbits can move horizontally or vertically (i.e., north, east, south and west directions), but not diagonally.
- In each generation, rabbits attempt to move themselves to an empty adjacent cell. If there are many empty adjacent cells, they choose one accordingly to a common criteria for selecting adjacent cells (see below). If there is no empty adjacent cell, they stay in the same place.

- Rabbits can procreate whenever GEN_PROC_RABBITS generations have passed since they were born or since they last procreated. Whenever a rabbit reaches such age (to procreate) and makes a move, it leaves in its last position a new rabbit and both rabbits' procreation age is set to zero.

Rules for Foxes

- Foxes can move horizontally or vertically, but not diagonally.
- In each generation, foxes try to eat a rabbit by moving to an adjacent cell that is occupied by a rabbit. If there are many adjacent cells occupied with rabbits, they choose one accordingly to the common criteria for selecting adjacent cells (see below). If there is no adjacent cell occupied by a rabbit, foxes attempt to move themselves to an empty adjacent cell by following the same criteria. If there is no adjacent cell occupied with a rabbit or empty, they stay in the same place.
- Foxes starve and die whenever GEN_FOOD_FOXES generations have passed since they were born or since they last ate a rabbit. Foxes die after not finding a rabbit to eat and before attempting to move themselves to an empty adjacent cell.
- Foxes can procreate whenever GEN_PROC_FOXES generations have passed since they were born or since they last procreated. Whenever a fox reaches such age (to procreate) and makes a move, it leaves in its last position a new fox and both foxes' procreation age is set to zero.

Rules for Rocks

- Rocks do not move and no animal can occupy its space.

Criteria for Selecting Adjacent Cells

- By following the clockwise order, start numbering from 0 the possible P cells to where a fox/rabbit can move (adjacent north, east, south and west cells).
- Let G represent the current generation and (X,Y) represent the cell coordinates where a fox/rabbit is positioned in the ecosystem space, then the adjacent cell to be chosen is determined by (G+X+Y) mod P. Assume that the initial generation is number 0 and that the world origin is (0,0).

Conflict Resolution

In each generation, start by moving the set of rabbits and only after the set of foxes. When moving the set of rabbits/foxes, you should assume as if all the rabbits/foxes move at same time, i.e., the particular order by which the rabbits/foxes are moved should not affect the final result. However, during the movement within a generation, it may happen that several rabbits/foxes try to move to the same cells. When this happens, you should apply the following rules:

- When two or more rabbits move to the same cell, only the one with the older procreation age (GEN_PROC_RABBITS) is kept and all the other rabbits die.
- When two or more foxes move to the same cell, only the one with the older procreation age (GEN_PROC_FOXES) is kept and all the other foxes die. If there are two or more foxes with the same procreation age, we keep the least hungry fox (GEN_FOOD_FOXES).

Implementation

Start by implementing a sequential version of the application and only then develop a parallel implementation for shared memory environments. For that, you can use processes combined with shared memory segments or threads by using the Pthreads or OpenMP programming models.

For the parallel implementation, start with simple strategies, e.g. having each worker responsible for the moves of a fixed portion of the world. Note that at the end of each generation, it may be required to synchronize data with adjacent workers about foxes and rabbits that moved from the space of one worker to the space of another. You can also split the matrix horizontally, vertically or in both dimensions in a static way, although more efficient solutions should be adaptive to dynamically balance the workload among the executing workers.

Input and Output

The application to be developed must accept as input a file with the configuration parameters and the description of the initial state of the ecosystem. Such a description must obey the following pattern: the first input line has all the configuration parameters (including an integer N representing the number of objects in the initial ecosystem), followed by N lines each representing one object of the initial ecosystem. The configuration parameters are the following (by order):

- GEN_PROC_RABBITS - number of generations until a rabbit can procreate
- GEN_PROC_FOXES - number of generations until a fox can procreate
- GEN_FOOD_FOXES - number of generations for a fox to die of starvation
- N_GEN - number of generations for the simulation
- R - number of rows of the matrix representing the ecosystem
- C - number of columns of the matrix representing the ecosystem

- N - number of objects in the initial ecosystem

The objects in the initial ecosystem are given using the format OBJECT X Y where OBJECT can be ROCK, FOX or RABBIT, and X and Y are the coordinates where the OBJECT is positioned in the matrix. An input example is:

```
2 4 3 6 5 5 9
ROCK 0 0
RABBIT 0 2
FOX 0 4
FOX 1 0
FOX 1 4
ROCK 2 4
RABBIT 3 0
RABBIT 4 0
FOX 4 4
```

The application should write to the standard output the description of the final state of the ecosystem, following the same format as the one used for the input. For the given example, the output should be:

```
2 4 3 0 5 5 5
ROCK 0 0
ROCK 2 4
RABBIT 3 0
FOX 4 0
FOX 4 3
```

In more detail, for the given example, the sequence of generations is as follows:

```
Gen 0     Gen 1     Gen 2     Gen 3     Gen 4     Gen 5     Gen 6
-------   -------   -------   -------   -------   -------   -------
|* R F|   |*R F |   |* F  |   |*  F |   |*   F|   |*    |   |*    |
|F   F|   |   F |   |  F  |   |     |   |     |   |     |   |     |
|    *|   |F   *|   |    *|   |    *|   |    *|   |    *|   |    *|
|R    |   | R  F|   |F    |   |     |   | R   |   |R    |   |R    |
|R   F|   | R   |   |  R F|   |FRR  |   | F R |   | FF  |   |F  F |
-------   -------   -------   -------   -------   -------   -------
```

The input file must be read from the standard input and the results are to be printed to the standard output (click **here** to get a set of test examples).

Implementation Requirements

- Visual display showing the grid with all entities
- Statistics tracking (population counts over time)
- Adjustable parameters (initial populations, reproduction rates)
- Time controls (pause, speed up, slow down)

Expected Outcome

The simulation should demonstrate cyclical population patterns where fox and rabbit populations rise and fall in relation to each other, with rocks affecting local dynamics. We should be able to observe how changing parameters affects the ecosystem's stability.

For this assignment, the development team may consist of either a single student or up to three students working together. This flexibility is intended to accommodate different working styles and encourage collaboration in tackling the challenges of parallel programming.

Submit a PDF report detailing the implementations, along with a public repository on GitHub, GitLab, Bitbucket, or a similar platform. Additionally, you may include a video to complement the report.