COMP 201 – Spring 2024

Assignment 2: Heap Management

Cycle of Life

Assigned 14 March 2024, Due: 28 March 2024 23:59

M. Burak Kizil (*mkizil19@ku.edu.tr*) is the lead person for this assignment.

Cycle of Life

In the boundless stretches of the cosmos, a new world has been discovered, teeming with the potential for life. This world, named Eden, holds the secrets of a self-sustaining ecosystem within its landscapes. "Cycle of Life" is a captivating zero-player simulation game that players initiate and observe the evolution of life in an untouched world. Through the lens of cellular automaton, the game offers a glimpse into the balance of nature, where life thrives under simple rules.

As the simulation commences, Eden is a blank canvas, a grid of possibilities. The players'

role is setting the initial conditions and then stepping back to watch the intricate dance of life unfold. The world of Eden is populated by three primary life forms: the resilient Plants, the roaming Herbivores, and the fierce Carnivores. Each species plays a critical role in the tapestry of life, locked in a perpetual struggle for survival, growth, and dominance.

Plants, the foundation of the ecosystem, spread across the barren lands, converting the sun's energy into life-sustaining nourishment. Herbivores, in their quest for survival, seek out the lush foliage, grazing on the plants to sustain themselves and reproduce. However, life in Eden is not without its perils, as Carnivores prowl the landscape, preying on the Herbivores in a vivid display of the food chain's cruel efficiency.

1 Introduction

In this homework, you will implement a zero-player simulation game following some rules and patterns.

Our goal is to read the given text files which contain the initial grid world with plants' herbivores' and carnivores' locations, simulate the game according to the rules described below, and print the simulation along the way.

Logistics

This is an individual assignment, and all hand-ins are electronic via GitHub. Any clarification or correction will be announced on the Blackboard.

Handout Instructions

How to Start

- Accept the GitHub Classroom assignment using the link:
  `https://classroom.github.com/a/LDmjWvSn`
- Clone the GitHub repository created for you to a Linux machine in which you plan to do your work (Using Linux servers [linuxpool.ku.edu.tr]. See **How to use linuxpool.ku.edu.tr linux servers** section for details.):

  ```
  $ git clone https://github.com/COMP201-Fall2023/assignment-2-heap-management-USER.git
  ```

  (Replace USER with your GitHub username that you use to accept the assignment)
- [**IMPORTANT**] After cloning the repository, you are required to write the following honor code in a new file called "honor.txt" and commit & push it: "I hereby declare that I have completed this assignment individually, without support from anyone else." You can use the following command to create "honor.txt" file and write the honor code in it:
  `$ echo "I hereby declare that I have completed this assignment individually, without support from anyone else." > honor.txt`
- Then, start filling in the `main.c` file. There is a bare-bones template ready for you to use. If you do not want to use this template, it is fine. The only necessities are clean, not spaghetti code, and the correct outputs.
- You can compile your code with the following code. Use of `-std=gnu99` is **highly** suggested.

  ```
  $ gcc -std=gnu99 -g main.c -o main
  ```

Task

Fill in Your Information

Please fill the comment at the start of the `main.c` file as following:

```
// Write your full name: YOUR_NAME, write your KU ID: YOUR_ID
```

2 Rules

Grid system

1. There are several maps under the maps folder. For each map, first line gives the width (column number) and second line gives the height of the grid (row number).
2. You need to dynamically allocate memory for the grid. It is **mandatory** to dynamically allocate temporary grids and use them for simulation. It is also important to clean up unused grids and prevent any memory leaks.
3. Points are zero-indexed from the top left corner. Thus, the top left corner is (0, 0), bottom right corner is (Height-1, Width-1).
4. Dashes (-) stands for empty cells. P stands for Plants, H for Herbivores and C for Carnivores. A cell can only be one of these four elements. During the simulation, you should make necessary adjustments to maintain this condition (i.e. checking for collisions).

Movements

1. Plants: Plants grow up by 'X' shape. If there is a plant at coordinates (r,c), in the next step (r-1, c-1), (r-1, c+1), (r+1, c-1), and (r+1, c+1) cells will be turned into plants **if they are empty**.

2. Herbivores: At each simulation step, a herbivore would check all adjacent cells to itself (up, down, right, left, up-right, down-left, etc.). **If there are no plants nearby, it will unfortunately die and will be turned into empty space**. If there are any plants, then herbivore will live and in addition, any plant cell that is nearby, will be turned into Herbivore cells due to reproduction. New born Herbivores do not eat any plant or reproduce at the same step they are born. This is also why we need to use two grids (one temporary) to control the simulation.

3. Carnivores: Carnivores similarly check all adjacent cells and if no Herbivore is found, then they will die. If there is at least one Herbivore nearby, then the Carnivore will live and in addition:

1- If there is a herbivore in their adjacent up, right, down and left cells, the Herbivore will die and leave its place to a newborn Plant.

2- If there is a herbivore in their adjacent corners, the Herbivore will die again but leave its cell to newborn Carnivore.

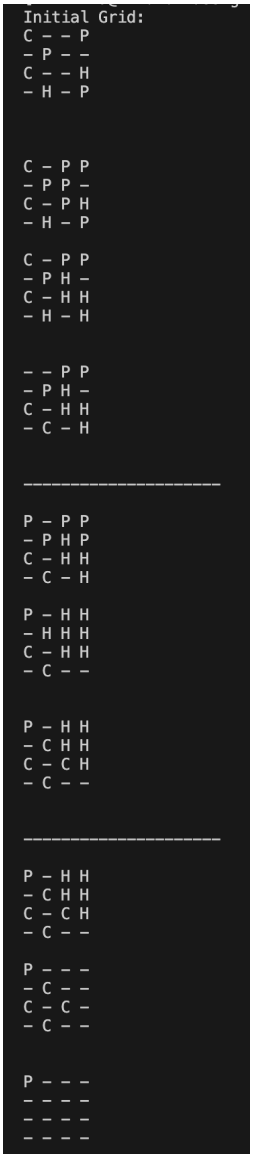
Simulation

At each step of the simulation, you need to execute the movements explained above in the following order. You can use temporary grids at calling each movement as long as you don't forget the clean-up the grids and free the allocated memory.

You should call the simulation using following command:

```
./output maps/map_n.txt num_steps
```

Example simulation with a few steps:



Evaluation

You are not allowed to use static structs. Your implementation must utilize dynamic memory allocation and structs.

Your score will be computed out of a maximum of 100 points based on the following distribution:

- **80** Correctness points
- **10** Effective use of version control points
- **10** Style points

Correctness points: Your code will be evaluated based on a set of different test maps and number of simulation steps and your outputs after each simulation step.

Effective use of version control points: You are required to push your changes to the repository frequently. If you only push the final version, even if it is implemented 100% correctly, you will lose a fraction of the grade because you are expected to learn to use Version Control Systems effectively. You do not have to push every small piece of change to GitHub, but every meaningful change should be pushed. For example, each of the functions coded and tested can be one commit. **For each function, there should be at least one commit (with proper commit message) that includes just modifications on that function.**

Style points: Finally, we've reserved 10 points for a subjective evaluation of the style of your solutions and your commenting. Your solutions should be as clean and straightforward as possible. Your comments should be informative, but they need not be extensive.

Handin Intructions

Same as previous assignments, we use GitHub for the submissions as follows. Note that we want you to get used to using a version management system (Git) in terms of writing good commit messages and frequently committing your work so that you can get the most out of Git.

- Commit all the changes you make:

```
$ git commit -a -m "commit message"
(Note: please use meaningful commit messages.)
```

- Push your work to GitHub servers:

```
$ git push origin main
```

Advice

- You will find many hints in the template code. While none of them is required, they are strongly advised since they both will make the assignment easier and will make it easier to get you partial points.
- Keep your GitHub repository by frequently committing the changes.
- Don't forget GDB and Valgrind since they can save a lot of time in debugging.
- Use linuxpool.ku.edu.tr Linux servers to test your code in order to avoid compatibility issues.
- You can write a makefile for your own project that helps you compile and run your code faster.

How to use linuxpool.ku.edu.tr linux servers

- Connect to KU VPN (If you are connected to the KU network, you can skip this step.) See for details:
- Connect to linuxpool.ku.edu.tr server using SSH (Replace USER with your Koc University username):

```
$ ssh USER@linuxpool.ku.edu.tr
(It will ask your password, type your Koc University password.)
```

- When you are finished with your work, you can disconnect by typing:

```
$ exit
```

Your connection to the server may drop sometimes. In that case, you need to reconnect.

We advise you to watch the following video about the usage of SSH, which is used to connect remote servers, and SCP, which is used to transfer files between remote servers and your local machine:

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss the given assignments with your classmates, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the web material as everything on the web has been written by someone else. See Koç University - Student Code of Conduct.

Late Submission Policy

You may use up to 7 grace days (in total) over the course of the semester for the assignments. That is, you can submit your solutions without any penalty if you have free grace days left. Any additional unapproved late submission will be punished (1 day late: 20 % off, 2 days late: 40 % off) and **no submission after 2 days will be accepted.**

Regulations

- **Blackboard:** This is an individual assignment, and all hand-ins are electronic. Any clarification or correction will be announced on the Blackboard.
- **Cheating:** We use automated plagiarism detection to compare your assignment submission with others and also the code repositories on GitHub and similar sites. Moreover, we plan to ask randomly selected 10% of students to explain their code verbally after the assignments are graded. And one may lose full credit if he or she fails from this oral part.