COMP 201 – Fall 2022

Assignment 3: Amogus

Heap, Dynamic Arrays, and Structs

Assigned 23 November 2022 23:59, Due: 7 December 2022 23:59

Osman Batur İnce (*oince22@ku.edu.tr*) is the lead person for this assignment.

Introduction

In this homework, you will code an adaptation of the famous Among Us game.

Figure 1: A screenshot from **Among Us** game

In a spaceship of X astronauts and Y impostors, the astronauts strike back! They have sabotaged the spaceship on purpose and caused a malfunction in the thermal control system and water pipe pressure, which resulted in ship-wide fog and pipes blowing hot steam. The fog affects only the impostors, reducing their sight to only vertical and horizontal neighbors, increasing the probability of an unseen astronaut neighbor being witness to the murder. The hot steam affects both the astronauts and the impostors, but differently. The hot steam forces the astronauts to move vertically first, and then horizontally, because of the pipe structure over the ship. But, the impostors cannot move at all. The astronauts have decided earlier that they will meet at the emergency room in case of a fog/wind sabotage, which is the center cell in a (2N+1)x(2N+1) grid world.

Now, the impostors are caught unaware, each one positioned in an independent cell.

The impostors will try to respond to this situation by killing every astronaut they can. In this process, if an impostor is seen by at least another astronaut killing the victim, then the impostor will be killed.

Keep in mind that we will only animate a single round the game, thus the result of every step is one of these three outcomes:
1. Defeat: The impostors managed to kill every astronaut, with at least 1 impostor remaining.
2. Victory: The astronauts managed to kill every impostor, with at least 1 astronaut remaining. Or, there are alive astronauts and alive impostors, and every alive astronaut made it to the emergency room.
3. Continue: There are alive astronauts and alive impostors, but not every alive astronaut made it to the emergency room.

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/eAApyvBW`
- 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-Fall2022/assignment-3-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 `among_us.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 among_us.c -o main
  ```

Task

Be careful that only alive impostors and astronauts can take action. For example, when we say *Every astronaut moves a single cell in a step*, we mean every ALIVE astronaut.

Fill in Your Information

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

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

which in my case results in:

```
// Write your full name: Osman Batur Ince, write your KU ID: 0084457
```

Grid system

1. $(2N+1)$x$(2N+1)$ grid world where $(N > 4)$. In the center, there is the emergency room that the astronauts try to go.
2. More than one person can be in the same cell, for example 2 astronauts and an impostor can be in the same cell. **Therefore, using dynamically allocated structs is required in this assignment.**
3. Points are zero-indexed from the top left corner. Thus, top left corner is (0, 0), bottom right corner is (2N, 2N), emergency room is (N, N), where the first value is the value of x-axis and the second value is the value of y-axis (**(x, y)**).

Sight

Impostors: Impostors can see only 4 of their neighbors, only vertical and horizontal neighbors and not the diagonal ones. Thus, they can only check these cells for a witness astronaut.
Astronauts: Astronauts can see all 8 of their neighbors, diagonal ones in addition to the vertical and the horizontal neighbors. Hence, there are 8 neighbors of an astronaut while there are only 4 neighbors of an impostor.

Movements

Each astronaut can move 1 cell at a time, which we will call step. However, the impostors cannot move at any step, they are **always stationary**. Therefore, the astronauts move a single cell each step. **Dead astronauts and impostors cannot move (what a surprise!)**.

Astronauts: The astronauts must get quickly to the emergency room, but the wind forces them to move vertically first, then move horizontally. So, if we take emergency room as the origin, first the astronauts must get to the x-axis vertically, then get to the emergency room horizontally.

Impostors: The impostors are stationary.

Rules

1. If an astronaut is not on the same cell with an impostor, the astronaut lives.
2. If an astronaut is in the same cell with an impostor:

2.1. If the impostor cannot see any neighbor of the astronaut (the neighbor can be in the same cell!), the impostor kills the astronaut.

2.2. If the impostor sees any alive neighbor of the astronaut (the neighbor can be in the same cell!), the impostor does nothing.

3. If an astronaut neighbor of an astronaut is killed, then the impostor that killed the astronaut is killed by the neighbor. However, the position of the killer astronaut **does not change**, the astronaut kills from range.

Input Format

The information in parantheses are only for information, they are not present in the actual input file. The astronaut count and impostor count specifies how many astronaut and impostor there will be respectively. N is the dimension parameter, which will be used to generate a $(2N+1)$x$(2N+1)$ grid world.

Next lines determine the astronaut and impostor coordinates. As you have to do two split procedures, you might want to look at the `strtok_r|` function.

The format of input are given in more detail in the `in_out_files` folder.

```
X (Astronaut count)
Y (Impostor count)
N (Dimension parameter)
S (Step count)
2,3 & ...  & 3,7 (Astronaut coordinates)
3,5 & ...  & 5,7 (Impostor coordinates)
```

Output Format

The information in parantheses are only for information, they are not present in the actual input file. `#DeadImpostors` mean the number of dead impostors, so it is only numeric. Moreover, think of `Y - #DeadImpostors` as a mathematical expression. So evaluate this expression and put the resulting value.

The format of output are given in more detail in the `in_out_files` folder.

```
#DeadImpostors (Dead impostors)
Y - #DeadImpostors (Alive impostors)
#DeadAstronauts (Dead astronauts)
X - #DeadAstronauts (Alive astronauts)
Victory (Status - One of {Defeat, Victory, Continue})
2, 3, Alive & ...  & 3, 7, Dead (Astronaut status)
3, 5, Alive & ...  & 5, 7, Alive (Impostor status)
```

You should be careful that if the game status changes to `Victory` or `Defeat`, the game ends. Therefore, the game can end before passing the step count specified in the input file.

UPDATE!

You should extract the input files into the same location as **among_us.c** file and read them from that location. Moreover, the output files, such as `input_text_1_out_0.txt`, should be written to the same directory as the **among_us.c** file.

```
assignment-1-USER
├─ ...
├─ input_text_1.txt
├─ input_text_1_out_0.txt
├─ among_us.c
└─ ...
```

Bonus Task (20 points)

If you want additional 20 points, you can do it by visualizing each step in ASCII format. You can put the illustration of each step in the following file format.

`input_text_0_out_7_vis.txt` where `input_text_0.txt` is the input file, currently we are at the **7th** step, and **vis** for visualization.

Note that multiple astronauts can be in the same cell, so you have to take that into account. Moreover, the visualization should be visually plausible and aesthetic. You cannot dump all the info and say that it is a proper visualization :).

Example

Run the generated binary executable on the `input_text_3.txt` file with the following command. In the illustrations, I, A, and O stand for `Impostor`, `Astronaut`, and empty cell respectively. You can look at info in the Task section to understand the steps better.

```
./output input_text_3.txt
```

The Tables 1-6 are just for illustrations, you do not have to make such visualizations unless you aim for the bonus points.

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	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	A	O	O	O	A
O	O	O	O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	I	O	O	O	O
O	O	A	I	O	O	ER	O	O	O	I	I	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	A	O
O	O	O	O	O	O	O	O	O	O	O	O	O
O	O	O	A	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	O	O	O	O	O

Table 1: Step 0 for `input_text_3.txt`. Note that no action has been taken yet in this illustration.

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	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	A	O	O	O	A
O	O	O	O	O	O	O	O	I	O	O	O	O
O	O	A	I	O	O	ER	O	O	O	I	I	O
O	O	O	O	O	O	O	O	O	O	O	A	O
O	O	O	O	O	O	O	O	O	O	O	O	O
O	O	O	A	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	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O	O	O	O

Table 2: Step 1 for `input_text_3.txt`

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	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	O	O	O	O	O
O	O	O	O	O	O	O	O	I	O	O	O	A
O	O	A	I	O	O	ER	O	O	O	I	O	O
O	O	O	O	O	O	O	O	O	O	O	O	O
O	O	O	A	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	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	O

Table 3: Step 2 for `input_text_3.txt`

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	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	O	O	O	O	O
O	O	O	O	O	O	O	O	I	O	O	O	O
O	O	A	I	O	O	ER	O	O	O	I	O	A
O	O	O	A	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	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	O
O	O	O	O	O	O	O	O	O	O	O	O	O

Table 4: Step 3 for `input_text_3.txt`

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	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	O	O	O	O	O
O	O	O	O	O	O	O	O	I	O	O	O	O
O	O	A	I	O	O	ER	O	O	O	I	A	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	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	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O	O	O	O

Table 5: Step 4 for `input_text_3.txt`

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	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	O	O	O	O	O
O	O	O	O	O	O	O	O	I	O	O	O	O
O	O	A	I	O	O	ER	O	O	O	I	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	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	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O	O	O	O

Table 6: Step 5 for `input_text_3.txt`

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
- **20** Bonus task (Visualize where people are in a separate file in a creative way (Don't forget that there might be multiple people at the same cell))

Correctness points: Your code will be evaluated based on a set of given input parameters and should result in correct numbers for:

1. Dead impostors and alive impostors
2. Dead astronauts and alive astronauts
3. *.txt files that demonstrate each step's status

You should pay extra attention to get the same output files in terms of name, white space, punctuation, and every single little detail. The autograder will grade your output files and otherwise you may lose significant amount of points.

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

- 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.

Sources

- Among Us Steam page