Curtin College Singapore
FOP1005 – Fundamentals of Programming

Assignment 2

30% of the unit Grading

Time Frame:

Release Date: 28th Aug 2023, 08:00 am (UTC+8)

Submission Date: 18th Sep 2023 08:00 am (UTC +8)

Contents and Percentage Distribution:

The details of important components of assignment and their constriction for the marking of this assignment are as below.

Sr.	Content	Percentage Distribution (if any)	Page No
1.	Overview	NA	
2.	Specification	NA	
3.	Timesteps	05%	
4.	Objects	10%	
5.	Movements	10%	
6.	Interactions	20%	
7.	Visualisation	05%	
8.	Parameter Sweep	10%	
9	Coding Standards	10%	
10.	User Documentation	20%	
11.	Report	10%	
12.	Submission instructions	NA	
13.	Requirements for passing	NA	
14	Lat submission	NA	
15.	Clarifications and Amendments	NA	

*All time details mentioned are in UCT+8 time zone standard
*Read the complete document and submission requirements before start working on it.
*This is an open book open computer/internet assessment however, any direct code copying form unacceptable links/repositories will be subjected as an academic integrity issue.

1 Overview

Deep in the jungles of Africa, a rare bat virus was transmitted to a group of people causing them to live forever without needing food or water. The minor side effect of this virus is that the infected people live with a continual desire to bite humans and drink their blood. Those bitten then join the world's new biting force (also known as vampires). This program is to simulate a vampire takeover and report on the findings.

2 Specifications

We will be simulating the infection of humans by vampires on a 2D map of a size of your choosing. The simulation will show a scatter plot containing humans, vampires, food, water, and garlic. On each timestep, the vampires and humans will move and interact with each other as well as with the food, water, and garlic

3 Timesteps (05%)

The heart of your program will be a loop that controls how many timesteps the program runs for. This loop will control all the other functionality (such as movement). Your program should accept 3 command line parameters but default to reasonable options in the case of their absence. The command line parameters are as follows:

- Initial number of humans
- Initial number of vampires
- Number of timesteps in the simulation

4 Objects (10%)

Both humans and vampires should be represented in the code as objects. Humans should have health and age variables and vampires only a health variable. They should also have methods for their various actions, such as moving, attacking, biting, etc. The methods you have are up to you. Humans start with a health of 100 at the beginning of the simulation and lose 1 health point for every step they move (see the movement section). Humans also start with a random age between 10 and 50 and at every timestep they age by 1. Humans don't live past 70 timesteps. Vampires start with the health that they had as a human before becoming infected. They don't have an age and only lose health by being bitten by other vampires (see the interaction section). Food, water, and garlic could also be represented using objects but don't have to be. The initial locations of food, water and garlic are up to you.

5 Movements (10%)

Humans and vampires both move randomly. Humans can move up to 4 spaces (steps) in any direction. Vampires can move up to 8 spaces (steps) in any direction. Neither humans nor vampires should go off the map. Food, water, and garlic don't move.

6 Interactions (20%)

Humans and vampires should interact with each other as well as items on the map (food, water, and garlic). An interaction occurs when a human or vampire is on any of the 8 squares next to something else (e.g., a human, a vampire, food, water, or garlic). When an interaction occurs, you should print the interaction and the result to the terminal (i.e., a vampire was killed by a human, a human just drank water, etc).

The human and vampire interaction types are as follows:

- **Human and human interaction**: some people are selfish, and others are helpful. When two humans interact there is a 40% chance that one human uses the other to gain 20 health, which means the other loses 20 health. There is also a 60% that the humans help each other, and both gain 10 health.
- **Human and vampire interaction**: vampires like biting. When a human and vampire meet the human has a 70% chance of being bitten by a vampire and becoming a vampire. The vampire has a 30% chance of being killed by the human.
- **Vampire and vampire interaction:** vampires like biting, so much so, they can't help but sneak in a cheeky bite at any opportunity they get. As a result, when vampires come into contact, they bite each other, with the result being that both vampires lose 20 health.

The water, food and garlic interaction types are as follows:

- **Human water interaction**: gains 50 health (no effect on vampires)
- **Human food interaction:** gains 30 health (no effect on vampires)
- **Human garlic interaction**: gains 100 health (no effect on vampires)

7 Visualisation of results (5%)

Your program should display a scatter plot for each timestep with the current positions of all the humans and vampires. The items on the map should vary in colour, shape, and size. For example, humans could be green circles, vampires' red circles, food brown squares, water large blue squares and garlic grey triangles. It is up to you what colours, shapes, and sizes you use.

8 Parameter sweeps (10%)

Create a parameter sweep to run the program with an initial human population of 10 to 40 with an initial vampire population of 1 to 5. Save the final plot of each simulation run by the parameter sweep. Also, save the initial and final human and initial and final vampire populations (4 values) of each simulation as one row in a CSV file (running the parameter sweep bash file should just produce one CSV file). You can use this data in your report.

9 Coding Standard (10%)

Your code submission must conform to coding standards emphasised in the lectures and practicals. Your code should also make use of multiple functions to reduce code duplication. Your code should also follow the program layout structure on slide 52 of lecture 4. Note, your code won't look the same as the code on that slide, but the order of the sections should be the same. Remember, consistency is key!

10 User Documentation (20%)

Your User Documentation will be a minimum of 2 pages long, in pdf format and should include the following:

- An overview of each of your program's features.
- A guide on how to use your program.
- A discussion of your code, explaining the features you implemented, how you implemented them and why you implemented them the way you did.

The user documentation will be used and referred to by the programmers in the data analysis team at your 'workplace' to get an understanding of your code.

11 Report (10%)

Your report will be a mini paper that is 2-3 pages long, in pdf format and should follow the structure of a standard academic report. This is what will be used by the World Health Organisation to make decisions about overcoming the vampire takeover. As a result, this report should discuss the insights you found from your simulation. Such as, how the initial vampire populations affect infection numbers, if there are any situations where all humans don't become vampires, etc.

The required sections are:

- Abstract: Summarise your report's findings. Explain the purpose of the program, the questions you wanted it to answer and your outcomes/recommendations.
- Background: Discuss the purpose of the program and your choice of questions (see examples in the paragraph above).
- Methodology: Discuss how you went about gaining these insights from the program to answer your questions and why you chose to do it that way. Include commands, input files, and outputs – anything needed to reproduce your results.
- Results: Present the results of your analysis – include tables, plots and discussions.
- Conclusion and Future Work: Give conclusions and what further investigations could be done.
- References

You can find examples on google but do not copy and paste. Just use the examples as a reference.

12 Submission Instructions

You should submit a single file, which should be zipped (.zip). The file must be named FOP_Assignment2_<student id> where the <student id> is replaced by your student id ignoring the angle brackets. There should be no spaces in the file name; use underscores as shown. After creating the zip file, open it and make sure everything zipped correctly!

The file must contain the following:

- Your code. This means all the files needed to run your program. That includes input files used as part of the assignment if that is required to run your program.
- README file, including short descriptions of all files and dependencies, and information on how to run the programs.
- User Documentation and Report, as described above.

A signed and dated Declaration of Originality. This is available on Moodle and asks you to confirm that your work is your own. You can sign a hard copy and scan it in, or you can fill in a soft copy and digitally sign it. Make sure that your zip file contains what is required. Anything not included in your submission may not be marked, even if you attempt to provide it later. It is your responsibility to make sure that your submission is complete and correct.

13 Requirements For Passing The Unit

Please note, as specified in the unit outline, it is necessary to have reasonably attempted the assignment to pass the unit. Section 2.5 of the Unit outline explains the weightage associated

with the assignment, which is 30% of the overall unit. As a guide, you should be able to achieve around 30% for this assignment to pass the unit. Plagiarism is a serious offence. This assignment has many correct solutions so plagiarism will be easy for us to detect (and we will). For information about plagiarism, please refer to http://academicintegrity.curtin.edu.au.

In the case of doubt, you may be asked to explain your code and the reason for choices that you have made as part of coding to the unit coordinator. A failure to adequately display knowledge required to have produced the code will most likely result in being formally accused of cheating.

Finally, be sure to secure your code. If someone else gets access to your code for any reason (including because you left it on a lab machine, lost a USB drive containing the code or put it on a public repository) you will be held partially responsible for any plagiarism that results.

14 Late Submissions

As specified in the unit outline, you must submit the assignment on the due date. Acceptance of late submissions is not automatic and will require supporting documentation proving that the late submission was due to unexpected factors outside your control. See the unit outline for details as to the procedure for requesting that an assessment be accepted after the due date.

Also, note that IT-related issues are almost never a valid excuse. In the event that you submit your assignment late and are deemed to have a valid excuse, you will be penalised 10% (that is, 10% out of 100%, not out of what you would have received) per calendar day that you are late, up to a maximum of seven (7) calendar days. Any work submitted after this time will not be marked and you will automatically fail the unit. Note that if you are granted an extension, you will be able to submit your work up to the extended time without penalty – this is different from submitting late.

15 Clarifications And Amendments

This assignment specification may be clarified and/or amended at any time. Such clarifications and amendments will be announced on the unit's Moodle page. These clarifications and amendments form part of the assignment specification and may include things that affect mark allocations or specific tasks. It is your responsibility to be aware of these by monitoring the Assignment section of the unit's Moodle page.