LESSON PLAN – Computer Programming

Title: Hobbits vs. Nazgul – Launch

Essential Questions	Can we model life-like behavior with Python? How can we apply our expanded understanding of data types and storage?

Learning Objectives :	1. Students will be able to explain the movement, reproduction, and death of Hobbits and Nazgul. 2. Students will be able to develop a group plan for administration and self-monitoring.

Standards (CSDF)	
9-12.CT.4	Implement a program using a combination of student-defined and third-party functions to organize the computation.
9-12.CT.5	Modify a function or procedure in a program to perform its computation in a different way over the same inputs, while preserving the result of the overall program.
9-12.CT.7	Design or remix a program that utilizes a data structure to maintain changes to related pieces of data.
9-12.CT.8	Develop a program that effectively uses control structures in order to create a computer program for practical intent, personal expression, or to address a societal issue.
9-12.CT.9	Systematically test and refine programs using a range of test cases, based on anticipating common errors and user behavior.
9-12.CT.10	Collaboratively design and develop a program or computation artifact for a specific audience and create documentation outlining implementation features to inform collaborators and users.

Teaching Materials:

Student handouts & Rubric(attached)
Graph paper

Procedure:

I. Forming Teams
   A. Students will select their team members
   B. Students will establish group norms and decide on a method of self-administration
II. Learning the Game
   A. Students will read through rules on handout & rubric
   B. Students will model enough generations in a small 5x5 grid on paper to develop understanding of rules
   C. Students will answer questions on handout to assess and demonstrate understanding of rules.
   D. Students will work on a project proposal during work time/as homework - this is a freeform text document, see assignments for an example with feedback.

Cellular Automata – Hunter and Prey

The goal for this programming project is to create a simple 2D predator–prey simulation implemented as a cellular automata. In this simulation, the prey are Hobbits, and the predators are Nazgul. These beings live in a fictional 2D world composed of a 30 X 30 grid of cells. Only one being may occupy a cell at a time, and the Hobbits are constantly on the run from the voracious Nazgul who like nothing better than to feed on the plump, un-muscled, succulent Hobbit flesh. Around the edge of this world there is an ominous city wall. Due to the casting of some ancient black magic the Nazgul are not capable of crossing this wall. They are essentially repelled and must change direction. On the other hand, the Hobbits, who appeased the witches with hand crafted ales and excellent long bottom leaf, are capable of ascending these walls and tele-porting to the other side of town. When a Hobbit approaches a particular edge it wraps around to the opposite edge and appears there, similar to the mythological beast Pac Man. As this world is an automata, it does not require any human interaction. It is simply seeded with life and set on its inevitable path. As the seeding is random there are infinite scenarios that can play out. Time in this world is simulated in discrete steps which can be implemented as a loop iteration. A single iteration over the entire world is equivalent to a 24-hour Earth day. Each being performs some action every time step. While each species is of the same genetic lineage and share the same basic behaviors (Move and Reproduce), the implementation is vastly different.

Hobbits:

- Move
    - A Hobbit will randomly attempt to move up, down, left, or right. If the cell in the selected direction is occupied, it will try another direction. If all four directions are occupied the Hobbit does not move.
    - A Hobbit can move "across the wall" and end up on the opposite side of the grid.
- Reproduce
    - If a Hobbit survives for three "days," then at the end of the third day (after moving) the Hobbit will asexually reproduce. A new Hobbit will be spawned in an adjacent empty cell (up, down, left, or right). If none of these four cells is empty, no new Hobbit will be produced.
    - Once an offspring is produced, this particular Hobbit cannot produce another until three more time steps have elapsed.

Nazgul:

- Move
    - If there is a Hobbit in an adjacent cell (up, down, left, or right), the Nazgul will move to that cell, stab the Hobbit with its Ringwraith sword, and consume it.
    - If there are no adjacent Hobbits, the Nazgul moves randomly like a Hobbit. Nazgul cannot cross walls to move to the opposite side of the grid.
- Reproduce
    - If a Nazgul survives for eight "days," then at the end of the eighth day (after moving) it will spawn a new Nazgul like the Hobbits do.
- Starve
    - If a Nazgul has not eaten a Hobbit within the last three "days," then at the end of the third day (after moving) it will starve and die. It should then be removed from the grid.

To understand the game better, you will play a few rounds on paper on a smaller grid.

1. Draw a square around a 5x5 grid on your graph paper. Label it "Gen 0."
2. Randomly assign starting locations of Nazgul and Hobbits:
    a. Represent Nazgul with an x and Hobbits with a dot.
    b. Roll the 8-sided die you were given for each cell
    c. 1-3 gives an empty cell, 4-7 gives a Hobbit, and 8 gives a Nazgul.

For each round:

1. Enclose another 5x5 grid on your graph paper and label it with your next generation. The first time, it will be "Gen 1."
2. Copy your current grid's contents lightly in pencil.
3. Traverse the board from left to right through each row. If the cell has a creature in it, take the appropriate actions for the creature type.
    a. When moving at random, use the 4-sided die to choose direction – 1 is up, 2 is left, 3 is right, and 4 is down.
    b. A Nazgul will always move to the first Hobbit it notices. It should check each direction in order until it finds a Hobbit and moves to it; if it does not, it should move randomly.
    c. You will need to track the number of moves since reproduction (or since spawn) for all creatures; do this by writing this number as a subscript on the creature's symbol.
    d. Nazgul also need to track number of moves since the last time they ate a Hobbit. Use a superscript for this.

Repeat this process for at least 8 turns, or until one of your populations dies out (whichever comes first).

Questions:

1. Give at least one way that you could simulate the rolling of a d4 to randomly choose a direction in Python.

2. Below are some Hobbit coordinates and a direction they are randomly chosen to move. Assume that the given direction is a free space. What are the new coordinates of the Hobbit after moving?

    (2, 3) right

    (1, 0) down

    (1, 0) left

    (4, 4) down

    (3, 1) up

3. Considering your answers above, consider writing a function for the "random move." How can you update coordinates for the edge case (when the Hobbit teleports to the other side)?

Project Rubric:
Life Game

CATEGORY (G for group grade, I for individual)	*Exemplar* 4 points	*Sufficient* 3 points	*Developing* 2 points	*Lacking* 0 or 1 point
(G) Programming - Concept & outline (Day 1)	Well reasoned concept, and final instructions meets spec	Concept & outline does not meet complete spec *or* final code either does not meet spec or can not be linked to concept	Concept & outline does not meet complete spec *and* final code either does not meet spec or can not be linked to concept	(0) Student group does not round out concept, does not meet instructor approval, no understanding of goal and no attempt to clarify
(G) Time & Accountability Tracking	Work is tracked in it's completeness, and it is clear who did what when	Work tracking has gaps, lacks reasonable clearness, but is otherwise well done	Work tracking lacks enough detail to reasonably discern reasonably who did what or is incomplete	(0) Group does not turn in or refuses to implement a system of tracking individual work tracking
(G) Code: Pred & Prey functionality & generalization *DOUBLE POINTS*	Entities are encapsulated in a dictionary or list, *and* interact with outside functions in a consistent way	Entities are encapsulated in a dictionary or list, *or* interact with outside functions in a consistent way	Entities are encapsulated in a dictionary or list in part and/or with major inconsistencies in structure between types	(0) Living entities are not modeled in any way per spec, cells are just text moving around, or are defined in the grid level
(G) Code: Functionalization, code reuse, & encapsulation	the main body of the file contains nothing other than function calls, global vars, and the game loop	Functions are used on levels comparable with the previous projects.	Functions are used in some places, but large portions of functionality are not in functions. Functions are used poorly (EG… 20 question functions vs 1 generalized question function)	(1) Code was not largely lacking in use of user defined functions
(G) Code: Functionality - game loop	Code works per specs	Code has minor issues but works mostly as intended	Code has significant issues, but attempts to approximate intent	(0) Code does not work as intended nor in any approximation of intent or code is not students own in any measurable way
(I) Code: Comments, spacing, & clarity	Commenting is on the block level & is clear, *and* spacing & tab use is clear & consistent	Commenting is on the block level or finer, but is vague *and* there are inconsistencies in spacing & tab use	*Either* comments *or* line spacing are not present at all and the other is below expectations or inconsistent *or* both are present but in no organized ways	(0) Neither comments or spacing are used in a clear way to help with code readability.
(I) Work Ethic: personal accountability	Work ethic is scored 0 - 8 based on instructor observation(2), work shown & log upkeep(2), and group/self feedback(4).			
Points Accumulated				

Total points **Possible Pts: 40** **Grade / 40**

Day 1 Checklist: Project Launch

You must have the following tasks accomplished before the end of class today.

Accomplished	Task
●	Select the members of your team whom you wish to collaborate with.
●	Establish a set of team norms and expectations for your team. Consider each of the following questions: ● How should the team each member responsible for the work that needs to be done complete the project? ● What should the consequences be if one (or more) team members fails to fulfill the expectations established by the team? ● How should the team move forward in case one (or more) members are out?
●	Read through the rules on the handout provided to you at the beginning of class. ● What questions do you have about the project? ● Discuss amongst your team members

	what you think the handout is asking of you. ● Write a list of five things you need to start thinking about right away when starting this project.
●	● Create a 5x5 generation model grid on paper ○ Go through the process a number of times so you begin to understand how the code should work.
●	● After you brainstorm and go through some work on paper, answer the questions provided on the handout. This will demonstrate to me that you understand the general rules of the code.