CSCI 1100 — Computer Science 1 Homework 8
Bears, Berries and Tourists Redux: Classes

Overview

This homework is wo [illegible] rd your overall homework grade, and is due **Monday, December 4th, 20** [illegible] It has three parts. The first two are not worth many points, and may end [illegible] ey are mainly there to give you information to help you debug your solution. [illegible] 3_files.zip. and unzip it into the directory for your HW8. You will find d [illegible] outputs for each of the parts.

The goal of this assig [illegible] classes. You will be asked to write a simulation engine and use classes to encapsulate data and functionality. You will have a lot of design choices to make.

While we have done simulations before, this one will be more complex. It is especially important that you start slowly, build a program that works for simple cases, test it and then add more complexity. We will provide test cases of increasing difficulty. Make sure you develop slowly and test throughly.

Submission Instructions

In this homework, for the first time, you will be submitting multiple files to Submitty that together comprise a single program.

Please follow these instructions carefully:

Each of Part 1, Part 2 and Part 3 will require you to to write a main program: `hw8_part1.py`, `hw8_part2.py` and `hw8_part3.py`, respectively. You **must** also submit three modules per part in addition to this main file, each of which encapsulates a class. The first is a file called `BerryField.py` that contains your berry class, a file called `Bear.py` that contains your Bear class and a file called `Tourist.py` that contains your Tourist class.

As always, make sure you follow the program structure guidelines. You will be graded on good program structure as well as program correctness.

Remember as well that we will be continuing to test homeworks for similarity. So, follow our guidelines for the acceptable levels of collaboration. You can download the guidelines from the resources section in the Course Materials if you need a refresher. We take this very seriously and will not hesitate to impose penalties when warranted.

Getting Started

You will need to write at least three classes for this assignment corresponding to a BerryField, a Bear and a Tourist. We are going to give you a lot of freedom in how you organize these three classes, but each class must have at least an initializer and a string method. Additional methods are up to you. Each c[illegible] bed below.

BerryField

The berry field must [illegible] the location of berries as a square Row X Column grid with (0,0) being the [illegible] l (N-1, N-1) being the lower right corner. Each space holds 0-10 berry unit[illegible]

- The initializer class must, minimally, be able to take in a grid of values (think of our Sodoku lab) and use it to create a berry field with the values contained in the grid.

- The string function must, minimally, be able to generate a string of the current state of the berry patch. Each block in the grid must be formatted with the "{:>4}" format specifier. If there is a bear at the location the grid should have a "B", if there is a tourist the grid should have a "T", and if there is both a bear and a tourist the grid should have an "X". If there is neither a bear nor a tourist, it should have the number of berries at the location.

- Berries grow. The berry class must provide a way to grow the berry field. When the berries grow, any location with a value 1 <= number of berries < 10 will gain an extra berry.

- Berries also spread. Any location with no berries that is adjacent to a location with 10 berries will get 1 berry during the grow operation.

Bear

Each bear has a location and a direction in which they are walking. Bears are also very hungry. In your program, You must manage 2 lists of bears. The first list are those bears that are currently walking in the field. The second is a queue of bears waiting to enter the field.

- The initializer class must, minimally, be able to take in a row and column location and a direction of travel.
- The string function must, minimally, be able to print out the location and direction of travel for the bear and if the bear is asleep.
- Bears can walk North (N), South (S), East (E), West (W), NorthEast (NE), NorthWest (NW), SouthEast (SE), or SouthWest (SW). Once a bear starts walking in a direction it never turns.
- Bears are always hungry. Every turn, unless there is tourist on the same spot, the bear eats all the berries available on the space and then moves in its current direction to the next space. This continues during the current turn until the bear eats 30 berries or runs into a tourist.
- For the special case of a bear and a tourist being in the same place during a turn, the bear does not eat any berries, but the tourist mysteriously disappears and the bear falls asleep for three turns.

- Once a bear reaches the boundary of the field (its row or column becomes -1 or N), it is no longer walking in the field and need not be considered any longer.

Tourist

Each tourist has a lo[illegible] bears, you must someplace maintain a list of tourists currently in the field [illegible]ts waiting to enter the field.

- The initializer c[illegible] be able to take in a row and column location.
- Tourists see a b[illegible]in 4 of their current position.
- The string function must, minimally, be able to print out the location of the tourist and how many turns have passed since they have seen a bear.
- Tourists stand and watch. They do not move, but they will leave the field if
  1. Three turns pass without them seeing a bear they get bored and go home.
  2. They can see three bears at the same time they get scared and go home
  3. A bear runs into them they mysteriously disappear and can no longer be found in the field.

Execution

Remember to get hw8_files_F19.zip from the Course Materials section of Submitty. It has two sample input files and the expected output for your program.

For this homework all of the data required to initialize your classes and program can be found in json files. Each of your 3 parts should start by asking for the name of the json file, reading the file, and then creating the objects you need based on the data read. The code below will help you with this.

```
f = open("bears_and_berries_1.json")
data = json.loads(f.read())
print(data["berry_field"])
print(data["active_bears"])
print(data["reserve_bears"])
print(data["active_tourists"])
print(data["reserve_tourists"])
```

You will see that field in a list of lists where each [row][column] value is the number of berries at that location; the "active_bears" and "reserve_bears" entries are lists of three-tuples (row, column, direction) defining the bears; and the "active_tourists" and "reserve_tourists" entries are lists of two-tuples (row, column) defining the tourists.

Part 1

In part one, read the json file, create your objects and then simply report on the initial state of the simulation by printing out the berry field, active bears, and active tourists. Name your program hw8_part1.py and submit it along with the three classes you developed.

Part 2

In part two, start off the same by reading the json file and create your objects and again print out the initial state of the simulation. Then run five turns of the simulation by:

- Growing the be [illegible]
- Moving the bea [illegible]
- Checking on the [illegible]
- Print out the st [illegible]

Do not worry about the reserve bears or reserve tourists entering the field, but report on any tourists or bears that leave. Name your program `hw8_part2.py` and submit it along with the three classes you developed.

Part 3

In part three, do everthing you did in part 2, but make the following changes.

- After checking on the tourists, if there are still bears in the reserve queue and at least 500 berries, add the next reserve bear to the active bears.
- Then, if there is are still tourists in the reserve queue and at least 1 active bear, add the next reserve tourist to the field.
- Instead of stopping after 5 turns, run until there are no more bears on the field and no more bears in the reserve list, or if there are no more bears on the field and no more berries.
- Finally, instead of reporting status every turn, report it every 5 turns and then again when the simulation ends.

As you go, report on any tourists or bears that leave or enter the field. Name your program `hw8_part3.py` and submit it along with the three classes you developed.