Name: _____

The final exam must be completed in class and within the allotted class time. All answers are to be submitted electronically. You will submit your answers on Moodle within a single `zip` archive, named `lastname.zip`. **You may not communicate with other students during the exam.**

Introduction to Python

40 1. For this question, submit your answers to `intro.py`.

   (a) Write a function named `fizzbuzz(low, high)`. This function will iterate over all numbers between `low` and `high` (as a half-open interval including `low` but not including `high`). If the number is divisible by 3, print `Fizz`, if the number is divisible by 5, print `Buzz`, and if the number is divisible by both 3 and 5, print `FizzBuzz`. Otherwise, just print the number itself.

Exploring Python

30 2. For this question, submit your answers to `tank.py`.

   (a) Create a class called `Tank`. It should have an initialization method (`__init__`) that takes in a `name`, `health`, `attack`, `position`, and `rng` (range).

   (b) Create three tanks: `t1`, `t2`, and `t3`. You can name these tanks "T1", "T2", and "T3". Each tank will have a health of 100, an attack of 10, and a range of 12. The position of T1 is (0, 0). The position of T2 is (1, 5). The position of T3 is (20, 2).

   (c) Within this class, write a method called `fire(self, targets)`, that takes in a `list` of tanks and fires upon the closest (using Euclidean distance) `Tank`. It should subtract `attack` units from the target, but only if it's own health is greater than 20 (otherwise, the unit is too damaged to fire). If all of the units are out of range, then the attack will not be successful (i.e, do nothing). The health of a unit can never fall below zero, and a unit can never attack itself.

Movement

20 3. For this question, you will add your changes to `wander.py` in the `wander` directory. The `wander` algorithm is often used to implement zombie movement in games.

   (a) The current implementation of the `wander` function is `pass`, which means that it does not do anything. Modify the `wander` function so that the AI implements the wander algorithm using section 3.3.2 (p. 53) of the AI text. To support this functionality, the following must be performed:

   (b) The `Avatar` class has already been modified for you so that it now contains the property `rotation`, which is measured in units of radians (angle). That is, a rotation of 0 means that the avatar will go to the right, a rotation of $\pi/2$ means that

the avatar will go up, a rotation of π (or $-\pi$) means that the avatar will go left, and a rotation of $3\pi/2$ (or $-\pi/2$) means that the avatar will go down. Any other radian angle is also possible.

(c) As a starting point, you will first implement the function `randomBinomial()`, which performs the calculation in p. 54 of the text. In Python, you can generate a random number between 0 and 1 using `random.random()`.

(d) Within `wander`, you will create a variable called `maxRotation`. Set `maxRotation = 0.5`.

(e) The enemy's new heading (`enemy.rotation`) is then the current rotation plus a randomly generated amount that is the random binomial multiplied by the `maxRotation` (r). That is:

```
enemy.rotation = (enemy.rotation + randomBinomial() * maxRotation)
```

(f) To rotate this vector to the desired rotation, perform:

$$t = \begin{bmatrix} \cos r \\ \sin r \end{bmatrix}$$

(g) This is your new target (t), which you can pass into `enemy.update`. You should use the `numpy` versions of the `cos` and `sin` operations. (If you don't know how to call these, find them in the `numpy` documentation).

(h) You can verify your solution by manually running `python wander_test.pyc`, which should behave similarly (but not exactly, due to the use of random numbers) as your implementation.

Learning

10 4. For this question, you will modify `game.py` in the `learning` directory.

(a) The current implementation of hide and seek has the AI randomly select a location. This means that regardless of how the player plays, the AI will only find the player with probability $1/n$, where n is the number of locations.

(b) Since human players are known to be biased, it is useful for the AI to instead keep track of the player's history, and then visit that location based on how often the player has hidden in that location in the past.

(c) We will do so using a probability distribution function. Consider three locations $A = 0$, $B = 0$, and $C = 0$. Since we have no evidence at this point, we will randomly visit any of the locations until we find the player.

(d) Now consider three locations $A = 2$, $B = 3$, $C = 1$. This means that the player has been seen at location A twice, location B three times, and location C once.

(e) To decide where to visit first, we will decide the percentage of time the player visits each of these locations. Ignoring rounding errors and converting these to integers, for A, this is 2/6 (33%), for B this is 3/6 (50%), and for C this is 1/6 (16%).

(f) Now roll a random number between 0 and 100. If this number is less than 33, try location A. Otherwise, if this number is less than 83 $(33 + 50)$, try location B. Otherwise, try location C.

(g) Let us say that we search location A first and that the player is not found in that location. We now repeat the full calculation, except using only locations B and C.

Question	Points	Score
1	40	
2	30	
3	20	
4	10	
Total:	100	