Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

Name:

Functions

1. Submit your solutions to these problems in func.py. Given the functions

$$G(x, y, z) = x + 2y + z$$

and

$$a(t) = t^3 + 2(t+1)^2 - t$$

program the following:

 $\boxed{5}$ (a) Re-write G(x, y, z) as a Python function.

(b) Re-write a(t) as a Python function.

(c) Add the following test code at the end of your script to verify your results:

2. The Heaviside step function¹ can be written in terms of the Dirac delta function and is defined mathematically as:

$$H(x) = \int_{-\infty}^{\infty} \delta(s) \ ds$$

As a discrete function, this becomes:

$$H[n] = \begin{cases} 0, & \text{if } n < 0 \\ 1, & \text{if } n \ge 0 \end{cases}$$

That is, the function returns 0 when n < 0, and returns 1 otherwise. In it's discrete form, this function is sometimes called a "unit step function". Re-write this as a Python function unitstep(n) and add it to func.py.

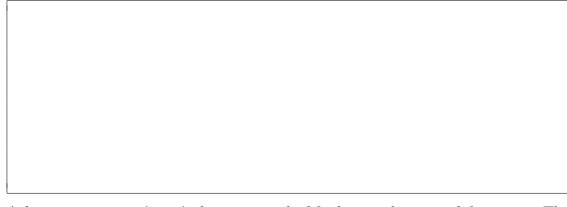
Distance Measures

- 3. Given two position tuples p as (x_1, y_1) and q as (x_2, y_2) , define the following functions in dist.py:
- (a) A function eucdist(p, q) that returns the Euclidean distance of the points. Recall that the distance function is

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{1}$$

¹https://en.wikipedia.org/wiki/Heaviside_step_function

10



(b) A function mandist(a, b) that returns the Manhattan distance of the points. The Manhattan distance is given as

$$d = |x_2 - x_1| + |y_2 - y_1| \tag{2}$$

where the vertical bars (|x|) represent the absolute value, whose Python function is abs. The game of chess uses Manhattan distance for rooks.



(c) Add test code to the script to ensure that both eucdist and mandist function correctly.

Classes

The code for this section should be submitted in a file called game.py.

4. Create an air unit called Bomber. A bomber has the following properties: name, position, health, attack, firing_range, accuracy, and air. Since a bomber is an air unit, air should always be set to True and does not have to explicitly specified when creating the Bomber. Thus:

bomber = Bomber("B1", (10, 10), 100, 6, 10, 70)

will create a bomber called "B1" at position (10,10) with 100 health, 6 attack, a firing range of 10, an accuracy of 70 (chance to hit), and having the ability to fly (that is, bomber.air will be True). Notice that air is not specified in the argument list.

- 5. Create a Tank class having the same properties as the Bomber, except that the air property will always be False.
- 6. Create a method called fire(self, targets) within the Bomber class. Since a Bomber drops bombs, the blast can damage multiple units at once; therefore, fire(self, targets) takes in a *list* of units, and not a single unit. However, the bomber can only attack ground units (that is, units with self.air = False), and so it can only attack Tanks, and not other Bombers.

For this problem, use Euclidean distance for the firing range. Accuracy is always between 0 and 100. That is, if the accuracy is 50, a Bomber only has a 50% chance to actually cause damage to the target, even if it is within firing range. Finally, a unit cannot cause damage to itself, and you can assume that every unit will be given a unique name.

7. Similarly, create a fire(self, targets) method within the Tank class. Unlike Bombers, Tanks can fire at both air and non-air units. Since Tanks can only fire at a single unit a time, it should pick the *closest* unit (that is within the firing range) in the list and attack only that unit.

To make this more concrete, let us make two bombers and three tanks:

```
b1 = Bomber("B1", (0, 0), 100, 6, 10, 100)

b2 = Bomber("B2", (0, 0), 100, 6, 10, 50)

t1 = Tank("T1", (3, 3), 100, 6, 10, 100)

t2 = Tank("T2", (1, 1), 100, 6, 10, 100)

t3 = Tank("T3", (2, 0), 100, 6, 10, 0)
```

In this example b1.fire([b2, t1, t2, t3]) will not cause any damage to B2, because B2 is an air unit. Because B1's accuracy is 100%, it will always cause damage to tanks T1, T2, and T3.

Now if we call t1.fire([t1, t2, b1]), then T1 will cause damage to T2 (but not B1), because T2 is the closest unit. It will cause any damage to itself, per the rules. Likewise, t3.fire([t2]) will never cause any damage, because T3's accuracy is 0%.

Question	Points	Score
1	12	
2	10	
3	17	
4	10	
5	10	
6	20	
7	20	
Total:	99	