Problem 1

Find the area of overlap between a heart (x^2 + (y – sqrt(|x|)^2 = 2) and a disk (x^2 + y^ 2 =2)

Python 3.9 was used for this program. This program can be run by running Test 1.1.py. I used Visual Studio Code to code and run.

Algorithm Description

I used Monte Carlo to find the overlapping region of the heart and the disk.

I first generated 100 million random coordinates from -2 to 2 for x and -2 to 3 for y.

```
for (repeated 100 million times:):
    x[index] = random float between [-2, 2]
    y[index] = random float between [-2, 3]
```

Now I check whether those coordinates are in both of the figures. The coordinate is inside both figures if the heart(x, y) and disk(x, y) are both <= 2.

```
for (repeated 100 million times):
    if heart(x, y) <= 2 and disk(x, y) <=2
        raise count by 1
ratio of count vs total = count/100 million
area of the region = 20 (since 4 * 5 = 2)
Area of overlap = 20 * ratio of count vs total
```

Results

The area I got is 4.136 units squared.

Performance

This program takes around 150 seconds to run. The fact that this program is in Python accounts for the long run time, as Python is a very slow programming language. Additionally, a lot of data points are calculated using some very expensive operations.

Problem 2

Find the two roots in [-1, 0] of the function f(x) = x^5 – 5x^4 + 5x^3 + 5x^2 – 6x -1

Python 3.9 was used for this program. This program can be run by running Test 1.2.py. I used Visual Studio Code to code and run.

Algorithm Description

Newton's Method was used to find the roots of this function.

```
def newtonMethod(x):
    if if |f(x)| < 0.00005:
        return x as the root
    else:
           use recursion and call newtonMethod with (x (f(x)/f'(x)))
```

In this function, I need to calculate f'(x). As a result, I created a function that calculates the derivative of f(x) using central difference. I used an h value of 1E-7.

```
def derivativeCD(x):
     return (f(x + h) - f(x - h))/(2h)
```

For Newton's Method, I need to input an estimate. My estimates for the two roots were -1 and -0.1

Results

The roots that I found were -0.9541 and -0.1510

Performance

Overall, I think that my program is able to solve the roots efficiently and elegantly. I think that using recursion in my functions makes it easy to see what the function is doing. The calculations in this program are very simple, and the program finishes almost instantaneously.