Analysis

1) To create the contour I used MatPlotLib's PyPlot module. This module uses a QuadTree and marching algorithm to create the contour. The description of the algorithm can be found here: https://github.com/matplotlib/matplotlib/blob/master/src/_contour.h
2) Answers to first four:
    1. When calculating the contour of the plot PyPlot stores each unconnected component in a paths list. The length of this list is the number of components.
    2. S is simply connected if all of the end points of its lines appear exactly twice. I used my module from program two to test this.
    3. PyPlot creates the contour so that positive is to the left of each line. SymPy's Polygon returns a signed area so if the area of the polygon is negative then the interior of the shape is empty.
    4. The max possible distance from my approximation to the boundary is *h* since the grid is sized so the diagonal of the box is *h*. Any feature larger than that would cross two sides of the box and be noticed.
3) One area where my algorithm makes assumptions is when there are more than one components in the graph area. Here instead of performing all of the tests on all of the components it only chooses the middle one and tests that. This was chosen because the individual components can have different answers to the questions and sometimes can have many pieces so the output became very complicated for the user (me!). I also chose to only check in a 10x10 grid. The program does not do any scaling to try and check if a shape is outside of the boundary or very small in some part of it.
4) The testing for sharp corners has some limitations. For the Cassini Oval which forms a figure of eight it says that there is a sharp corner because PyPlot creates the shape so that the positive side is always to the left of the lines. This means that in the middle instead of having the boundary cross it forms two coincident notches. The sharp corner algorithm then returns that it has a sharp corner. If the boundary was instead allowed to cross in the middle there would be not sharp corners.
5) Extra Credit
    1. Sharp Corner Test:
        1. Since angles wrap around (cross zero) trying to compare angles of lines did not seem like a good way to test if there were sharp angles in the shape. Instead I decided to normalize the length of each segment to one and then find its deltaX and deltaY, what I have called the normalized slope. Then all I compare the normalized slope of one line to the next and if either delta is > a tolerance there is a sharp corner. Due to some irregularities in the contour I found that taking the rolling average of a few lines in a row helps smooth out the data and provide better results. From testing a rolling overage of 2 or 3 seems sufficient. A tolerance of 0.4 is around 23 degrees and also seems to work.
    2. IsDistance
        1. Here I simply find a point that is outside of the shape and nearest to an existing point. I take the coordinates for that point and put it into the distance function. If the result is the same as the distance between the test point and shape point then the function is the distance function.