CS 3320

Module 12 – Programming Assignment

"Adaptive Integration"

Write a function, `area(func, a, b, tol)`, that computes the area between the curve of a function, `func(x)`, and the x-axis on the interval `[a, b]`. The function uses the adaptive Simpson technique described in pseudocode in the class notes. Use an error tolerance of 5.0e-6. Test your program on known integrals, and/or use `scipy.integrate.quad()` function to check your work. Then run your program on the following three integrals:

$$\int_{-1}^{1} e^{x^2}\,dx, \quad \int_{-1}^{10} \frac{\sin x}{x}\,dx, \quad \int_{0}^{1} \frac{\sin x}{x}\,dx$$

Try to minimize the number of function evaluations. My output for these three integrals with a tolerance of 5.0e-6 is shown by the following execution:

```
tol = 5e-06
(nevals = 49) e^x^2 [-1,1] = 2.9253035
(nevals = 97) sin(x)/x [-1,10] = 2.6044306
(nevals = 5)  sin(x)/x [0,1] = 0.9460830
```

The numbers in parentheses are the number of function evaluations (of `func`) performed for each complete call to `area()`. (I did some optimizations to get those low numbers—get things working first and then do the optimizations) Also, remember that $\lim_{x\to 0} \frac{\sin x}{x} = 1$.

To optimize the number of function calls, do *not* use a list or any other form of memorization. Instead, write a helper function (I called mine **simpson**) to which you pass not only the points *a* and *b*, but also pass the function values at *a*, *b*, and (*a*+*b*)/2. So **`area`** is really only called once, computes these starting values, and then passes 7 arguments to **`simpson`**, which then performs the calculation in the pseudocode for **area** in the slides. In this case, **`area`** is the main driver and **`simpson`** contains the recursive adaptive algorithm. The idea is to never call a function twice on the same *x*-value, while also avoiding the overhead of using an additional data structure. This is similar to the technique we did on minimizing the number of function-calls in the root-finding program, except in this case, we are passing intermediate values as function parameters instead of just assigning them to variables.

You *must* do this optimization for full credit.