COS126: General Computer Science	**Spring 1996**
Problem Set # 2	
Handed Out: Feb. 13, 1996	*Finish by: Feb. 21, 1996*

Let f be any function with one argument, such that $f(x) > 0$ for all x. For purposes of this assignment, f is given by a function in C with the following declaration.

```
double f ( double x);
```

The goal of this assignment is to write a C program that computes the area under the curve $y = f(x)$ in the interval from $x = l$ to $x = r$, where l, r are inputs to the program. (If you have taken calculus, you know that this area is also the definite integral of $f(x)$ from $x = l$ to $x = r$. However, you won't need to know calculus to do the assignment.)

When the function f is very complicated, this area can only be *estimated* (i.e., computed within some precision). Most methods for this estimation involve slicing the interval from $x = l$ to $x = r$ into n equal-sized intervals of width ϵ, where $\epsilon = (r - l)/n$. The number n is chosen by the user; the larger n is, the better the precision of the calculation. This partitions the area under the curve $y = f(x)$ into vertical strips of width ϵ. Then we estimate the area of each vertical strip, and add up the estimates for all the strips. Clearly, if the area of each vertical strip was estimated accurately, then this sum is an accurate estimate of the overall area.

How should we estimate the area of each strip? This assignment will explore several methods. One very simple estimate is (see Figure 1):

$$\text{strip area} \approx \text{width of the strip} \times \text{value of } f \text{ at midpoint of the strip.}$$

With this estimate, the estimate for the overall area would be

$$\text{area\_midpoint} = \sum_{i=0}^{n-1} \epsilon \cdot f\left(l + (i + \frac{1}{2}) \cdot \epsilon\right). \tag{1}$$

This can be computed using the following C function.

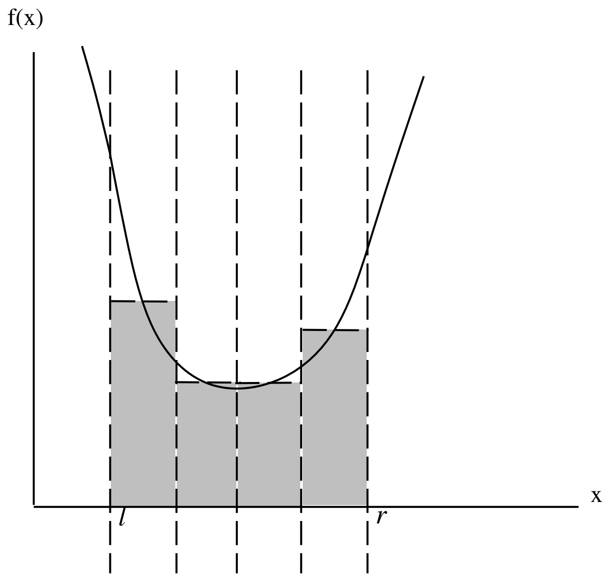

Figure 1

Estimating the area under a curve using the midpoint method.
Shaded area is the area computed.

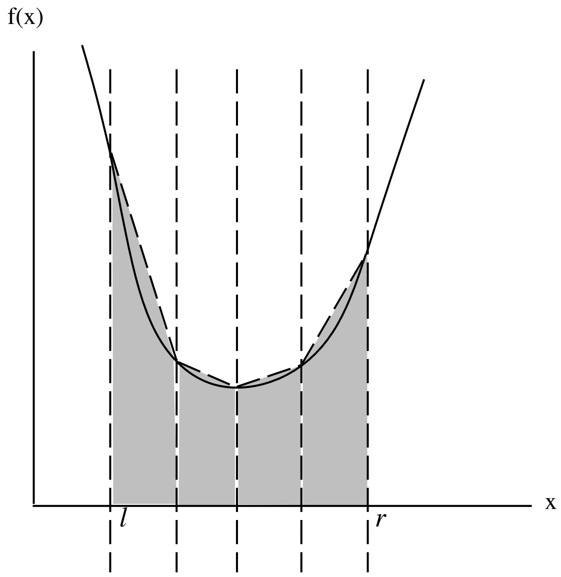

Figure 2

Estimating the area under a curve using the trapezoid method.
Shaded area is the area computed.

```
double area_midpoint (double l, double r, int n)
      {
        int i;
        double sum,  epsilon;

        sum = 0;
        if (n > 0)
            epsilon = (r - l)/n;
        else  return 0;
        for (i = 0; i < n; i++)
          {
            sum = sum + (epsilon *  f( l + i*epsilon + epsilon/2));
          }
        return sum;
      }
```

Another method, the *trapezoid method* (see Figure 2), approximates the area of the strip from $x = a$ to $x = a + \epsilon$ as

$$\text{strip area} \approx \text{width of the strip} \times \frac{f(a) + f(a+\epsilon)}{2}.$$

Thus the estimate for the area of all the strips is

$$\text{area\_trapezoid} = \sum_{i=0}^{n-1} \epsilon \cdot \frac{f(l + i \cdot \epsilon) + f(l + (i+1) \cdot \epsilon)}{2}. \tag{2}$$

Finally, a third way to estimate the area is to compute the above two estimates and take their average.

$$\text{area\_hybrid} = (\text{area\_trapezoid} + \text{area\_midpoint})/2 \tag{3}$$

Often, area_hybrid is a better estimate than either area_trapezoid or area_midpoint. For *convex* functions[1], the difference between area_hybrid and the actual area is at most (area_trapezoid – area_midpoint)/2. (You won't need to understand convex functions to do the assignment.)

Now we describe the programs you have to write. Note that the type "double" gives better precision than "float"; use it throughout this assignment. Also, make sure that your program outputs results in a nicely formatted fashion (i.e., with appropriate linebreaks and blank spaces).

Part 1, 50%: Write a C function that uses the trapezoid method to estimate the area under a function $f(x)$ in the interval from $x = l$ to $x = r$. To test your implementation of the trapezoid method, implement the function $f(x) = x^2 + 1$ and write a main procedure that prints a list of the estimated area using first the midpoint method and then the trapezoid method for the following intervals:

- $l = -1$ and $r = 1$
- $l = 0$ and $r = 2$.

Use a division into 100 strips ($n = 100$).

Now change your function to $f(x) = x^3$ and repeat all the above steps.

Part 2, 10%: Modify your main() procedure to compute area_hybrid for $f(x) = x^4$ and $l = -1, r = 1$ and for $n = 10, 100, 1000$. Your new main procedure should print a list of the results of this computation.

Part 3, 40%: In this part, the program will continue to use area_hybrid. However, the new twist is that the program has to decide upon a "good enough" n. The user specifies the maximum allowable difference between the true area and area_hybrid. The program then figures out a value for n to achieve this.

We will restrict attention to convex functions, For convex functions, an upper bound on the error made by area_hybrid is (area_trapezoid – area_midpoint)/2. Write a program that finds the smallest n for which this upper bound is less than .00001 (i.e. 10^{-5}). You should write your program to work for any convex function f and any interval $[l, r]$, but try it on $f(x) = x^4$ with $l = -1$ and $r = 1$. For these values, the

[1]A function f is said to be *convex* if for any a and b, the values of the function between $x = a$ and $x = b$ lie below the straight line joining $(a, f(a))$ and $(b, f(b))$.

correct n is between 100 and 1000. Note however that trying all values of n starting at 100 is very inefficient. Instead, use "binary search": Try $n = 550$, the middle value among the possible values for n. If that n gives too large an error, you know the n you want is in the range 550 to 1000; try a new value of n in the middle of that range. If $n = 550$ gives an error less than .00001, you know the n you want is in the range 100 to 550; try a new value of n in the middle of that range. Continue to halve the range of possible values for n with each try. Note that after the first round, your range of possible n may contain neither $n = 100$ nor $n = 1000$, so you need to keep track of the two ends of the range you are searching.

Submission: Put the four programs that you write in four separate files: (1) `part1sqr.c` with the program for Part 1 with $f(x) = x^2 + 1$; (2) `part1cube.c` with the program for Part 1 with $f(x) = x^3$; (3) `avg.c` with the program for Part 2; and (4) `binSearch.c` with the program for Part 3. To electronically submit the four programs, type

```
/u/cs126/bin/submit 2 part1sqr.c part1cube.c avg.c binSearch.c
```

Programs must be electronically submitted by **10pm on Wednesday, February 21, 1996.** Hardcopy must be turned in during lecture the next morning. Remember to use the official cover sheet for the hardcopy you submit.

Challenge for the bored: What if you need to find the area under a curve that is not convex? Then the averaging of area_midpoint and area_trapezoid may not be the best strategy. Read the chapter on integration in *Algorithms in C* by Sedgewick and implement one of the more sophisticated methods.