COS128: General Computer Science - Honors	Spring 1996
Problem Set # 1	
Handed Out: Feb. 6, 1996	*Finish by: Feb. 14, 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

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

This can be computed using the following C function.

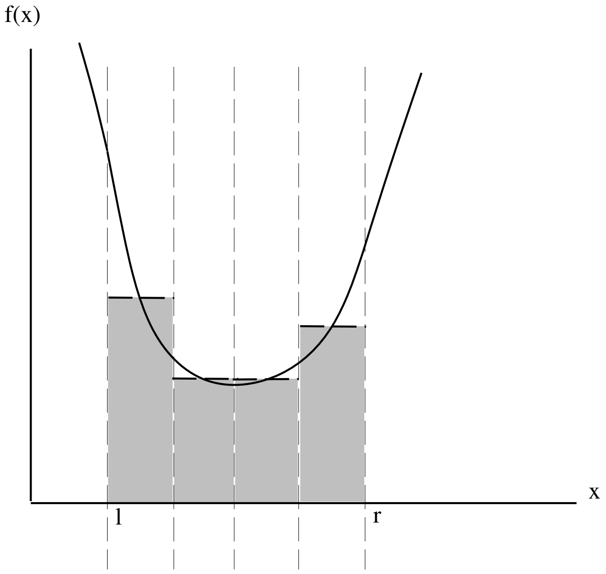

Figure 1

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

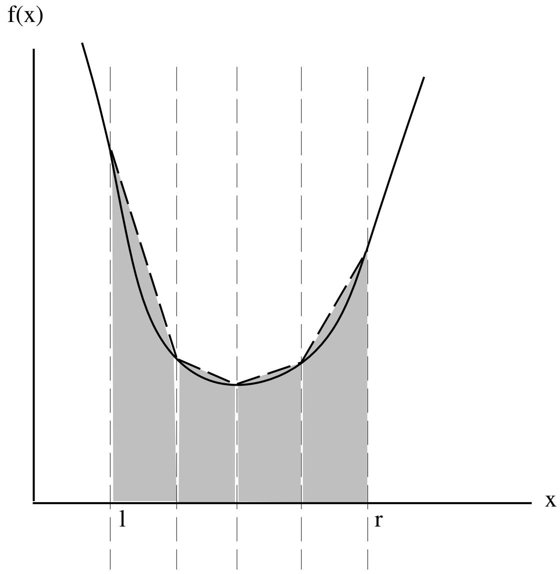

Figure 2

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 = 0,  epsilon;
        if (n > 0)
            epsilon = (r - l)/n;
        else  return 0;
        for (i = 0; i < n; i++)
          {
            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

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

Now we describe the programs you have to write. Note that the type "double" gives better precision than "float"; use it throughout this assignment.

Part 1, 15%: 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, 35% Combining the midpoint and trapezoid methods can give a better estimate. In particular, for convex functions,[1] using the estimate (area_midpoint + area_trapezoid)/2 gives an error no greater than (area_trapezoid – area_midpoint)/2. One class of convex functions are the functions x^k for even k. For these we have an exact equation for the area between l and r:

$$\text{Area} = \frac{1}{k+1} r^{k+1} - \frac{1}{k+1} l^{k+1} \tag{3}$$

For $l = -1$ and $r = 1$, this reduces to $2/(k+1)$ when k is even.

For part 2 of this assignment, write a main procedure that prints the area estimate given by (area_midpoint + area_trapezoid)/2, the error estimate (area_trapezoid – area_midpoint)/2, the actual area $2/(k+1)$, and the actual error [2] for function x^4 and $n = 10$, 100, and 1000 strips.

Part 3, 35%: Write a program that finds the smallest n for which the error estimate (area_trapezoid - area_midpoint)/2 is less than .00001 (i.e. 10^{-5}). You should write your program to work for any definition of the function f and any interval $[l, r]$, but try it on x^4 with $l = -1$ and $r = 1$. Your results from Part 2 should show that the smallest n is between 100 and 1000. Note 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 r; 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 l 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, you may cut off l and r, so you need to keep track of the two ends of the range you are searching.

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

[2] The use of the floating point representation also introduces error, because it is a finite precision representation (e.g. 1/3 cannot be represented exactly with finite precision decimals). Therefore, even the "actual error" may not be exact.

Part 4, 15%: Redo part 3 in the case when you aren't given a range for n. One way is to guess a modest value of n; if that value is not large enough (the error is still too high), double your guess for n. Continue doubling your guesses for n until you find an n that does give a small enough error; then turn to your "binary search" of Part 3. Test your solution on $f = e^x$ for interval $l = 0$ and $r = 1$ and error less than .0000001 (i.e. 10^{-7}).

Submission: Put the five programs that you write in five 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; (4) `binSearch.c` with the program for Part 3; and (5) `freeSearch.c` with the program for Part 4. To electronically submit the five programs, type

```
/u/cs128/bin/submit 1 part1sqr.c part1cube.c avg.c binSearch.c freeSearch.c
```

Programs must be electronically submitted by **10pm on Wednesday, February 14, 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.