CS 1713, Introduction to Computer Science
Assignment 5, Spring 1998

Root of an Equation

Due February 26, 1998

The objective of this programming assignment is to find a real number x satisfying $\cos(x) = x$, where $0 \leq x \leq \pi/2$. (Draw the graphs of the line $y = x$ and the parabola $y = \cos(x)$ to see that they cross at a point with x-coordinate between 0 and $\pi/2$, and this is the root we want.)

You are to try out *two different* root-finding methods for finding this x value, applying them to the equation $f(x) = \cos(x) - x$. (A root is a place where $f(x) = 0$, or $\cos(x) - x = 0$, or $\cos(x) = x$.)

1. Newton's method. The first approach will use *Newton's method*, which is taught in calculus courses. (Don't worry if you haven't had calculus, because you just need to use the formulas given below and don't need to know calculus.) For this method, one also needs the derivative of f, $f'(x) = -\sin(x) - 1$. Newton's method starts with a guess at the answer, call it x_0, and produces a new, hopefully better guess x_1 using the "magic" formula

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Replace the old guess with the new one and repeat in a loop until the old guess and the new guess are within some small value ϵ of one another. (You should ask if the absolute value of their difference is less than ϵ, using the function `fabs`.) In this assignment we will take ϵ to be 0.000000000001 . You should use the new guess as your final answer. Your program must use two C functions named "`f`" and "`fprime`" as shown below, so that the magic formula actually uses these functions. Finding the root must be done with a separate C function named "`newton`" with two input parameters, the initial guess x_0, and the tolerance ϵ. Thus the header for this function should look like:

```
void newton(double x0, double epsilon)
```

Use $x_0 = 0$ for the initial guess.

```
double f(double x)
{
   return cos(x) - x;
}

double fprime(double x)
{
   return -sin(x) - 1.0;
}
```

2. Bisection method. The second approach will use the *bisection method*. Here one starts with two values x_1 and x_2 (with $x_1 < x_2$) for which $f(x_1)$ and $f(x_2)$ have opposite signs (i.e., one is positive and the other is negative, or $f(x_1) * f(x_2) < 0$). If f is a continuous function (no jumps or breaks), then there must be an x between x_1 and x_2 for which $f(x) = 0$. To get closer to that value, let $x_{mid} = (x_1 + x_2)/2$, the midpoint. Consider $f(x_{mid})$ and replace either x_1 or x_2 by x_{mid}, so that after the replacement we still have $f(x_1)$ and $f(x_2)$ with opposite signs, but now x_1 and x_2 are half as far apart as they were before. Repeat this process until the distance between x_1 and x_2 is less than ϵ, again taken to be 0.000000000001 in this assignment. Your final answer should be x_{mid}. Again you must use a C function named "`f`". Finding the root must be done with a separate C function named "`bisection`" with three input parameters, the initial numbers x_1 and x_2 and the tolerance ϵ. This time the header for the function should look like:

```
void bisection(double x1, double x2, double epsilon)
```

Use the numbers 0 and $\pi/2$ as the initial values.

Directions for both approaches. Your program should first read the value 0.000000000001 for the number ϵ used to determine how close an approximation to the root one gets. (If you wish, you can instead give ϵ its value using a constant or a `#define`.) Then print "`Tolerance value:`" and the value for ϵ. For each of the methods the program must *count* the iterations and must terminate the loop in case the iteration count gets larger than 50. First you should print "`Newton's method:`" and a short list of approximate values obtained at each iteration, ending with a final approximation. Next print "`Bisection method`" and a somewhat longer list of *pairs* of values obtained at each stage of the second method, again ending with the final approximation. Notice that you should get (almost) the same answer by the two methods. All real numbers must be printed out with 15 digits to the right of the decimal place. Of course you must use a makefile and a separate directory for the files for this assignment.

Outline of the source file.

```
/* #includes, #define for PI */
/* function prototypes: f, fprime, newton, bisection */
void main(void)
{
     double epsilon;
     /* scanf for epsilon */
     printf("Newton's method\n");"
     newton(0.0, epsilon);
     printf("Bisection method\n");
     bisection(0.0, pi/2.0, epsilon);
     exit(0);
}
/* function definitions */
```