Computer Science 102
Lab 12

In this lab you will you will implement a C language function that can find numerically a value for which an arbitrary continuous function takes on a value of approximately 0. You should use the *bisection* algorithm that was described in the note in conjunction with surfaces of revolutions (revsurfs).

```
/**/
/* This function attempts to find a zero of a function */
/* using the bisection method                          */

int bisect(
double (*f)(double),    // external function whose zero we seek
double *minval,         // pointer to minimum of search range
double *maxval,         // pointer to maximum of search range
double epsilon);        // tolerance
```

Your function should begin by ensuring that *maxval > *minval and that f(*minval) and f(*maxval) have different signs. If not *bisect* should return 0.

Otherwise your function should perform bisection until

```
(*maxval - *minval) < epsilon
```

In this case the value returned should be the number of times the interval is bisected. Your bisection function should directly modify the values *minval and *maxval so that on return the caller can use *minval as the root of the equation f(t) = 0.

```
int main()
{
   int code;
   double min;
   double max;

   min = 0.5;
   max = 2.0;

   code = bisect(f1, &min, &max, 1.0e-10);
   printf("code = %5d min = %16.6le f(min) = %16.6le \n",
             code, min, f1(min));
```

In this lab you will submit a single file, bisect.c that includes *ONLY* the new bisect() function constructed as part of this lab. It must not contain the main() function nor the f1() function!

sendlab.102.labsection# lab# vector.h