

# Engineering C/C++ - EGR 155

**Instructor:** Harry McMaken

**Phone:** 1-800-362-2127 x6861,DMACC (long distance)  
964-6861,DMACC (local)

**E-Mail:** hlmcmaken@dmacc.edu

**Web Page:** [www.dmacc.edu/instructors/hlmcmaken](http://www.dmacc.edu/instructors/hlmcmaken)

**Office:** Building 4 Room 10C

**Hours:** 9:00-10:00am MTWRF or by appointment

## Course Policies:

Students are responsible for keeping current with daily assignments and completing all projects in a timely manner. There will be four programming projects assigned with point values of 10, 10, 10, and 20. The fourth project may be a team project approved by the instructor, or a program assigned by the instructor. Tests are to be taken on the day scheduled unless other arrangements are made prior to the exam. There are two exams. Exams are open book, open note. Each exam is worth 25 points. The final exam will be just a standard hour exam covering the last chapters of material. The students final grade will be the total of all exam and project scores based on the following schedule:

90 - 100 - A      80 - 89 - B      70 - 79 - C      60 - 69 - D      Below 60 - F

Additional help is available in the Learning Center, Building 6.

Date	Assignment		
Aug 27	Introduction & Data Structures	Read Chpt. 1	
Sep 3	Getting Started with C	Read Chpt. 2	
Sep 10	The Basics of C	Read Chpt. 3	
Sep 17	The Basics of C	Read Chpt. 3	
Sep 24	if and switch statements	Read Chpt. 4	Program 1 due
Oct 1	Loops	Read Chpt. 4	
Oct 8	Functions	Read Chpt. 5	Team Project proposals due
Oct 15	<b>Exam 1</b>		
Oct 22	Arrays and Index Variables	Read Chpt. 6	Program 2 due
Oct 29	Strings & Pointers	Read Chpt. 7	
Nov 5	Data Structures & Program Design	Read Chpt. 8	
Nov 12	Recursion	Read Chpt. 8	Program 3 due
Nov 19	Numerical Methods		
Dec 3	Numerical Methods & Review		
Dec 15	<b>Exam 2</b>	<b>Final Program due</b>	

## Program 1: Equations of a Line

Write a program to compute the equation of a line in slope intercept form given two points on the line. Your program should be able to handle any two points. The program should ask for the two points as input, and output the equation of the line.

## Program 2: Finding Roots using Secant Method

Consider the figure to the right. The point at which the curve crosses the x axis is call the root of the function. In many cases it is not possible to get an exact expression for the root of a function and you must obtain a numerical approximation to the root. One method for finding the approximation is called the Secant Method. The Secant method begins by selecting two values of x near the root and calculating the corresponding function values. A secant line is then constructed using the two points  $(x_1, f(x_1))$  and  $(x_2, f(x_2))$ . The x intercept for the secant line is calculated and is used as the new approximation to the root. A new secant line is constructed using the previous point and the current x intercept, and the process is continued until the root is determined within a given tolerance. This process may be summarized by the following formula:

$$x_{n+1} = x_n - f(x_n) \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

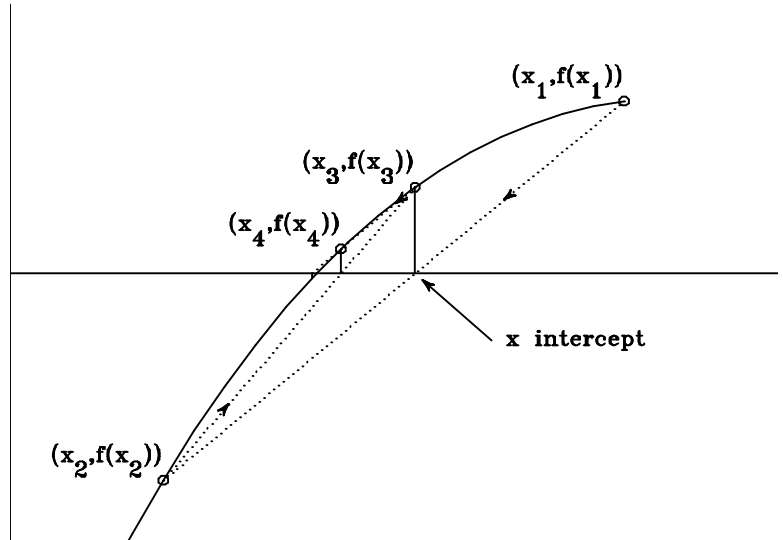
The Secant Method requires just one new function evaluation at each step.

Your assignment is to write a function subroutine which implements the secant method. The prototype for the function subroutine is:

```
float secant(float x1, float x2, float tol, float (*f)(float x))
```

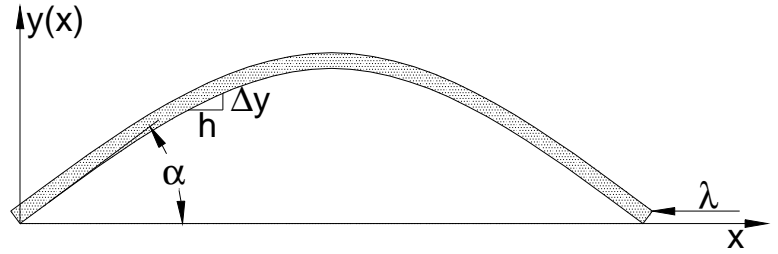
The routine should return the approximation to the root. The Secant routine must be independent of the specific function for which you are locating the root. Use this subroutine to find all three real roots of

$f(x) = x^3 - 3x + 1$   
to a tolerance of  $5 \times 10^{-5}$ .



### Program 3: Determining the Deflection of a Beam

This program will calculate the deflection in a beam subject to a compression load  $\lambda$  (see the figure to the right). The beam has been normalized so that the endpoints are 0 and 1. For small values of  $\lambda$  the beam will not deflect. As  $\lambda$  exceeds  $\pi^2$ , the beam will begin to deflect. The program will consist of five parts:



1. A main module,
2. A one step differential equation solver (provided),
3. A forcing routine for the differential equation solver,
4. A root finder,
5. A routine to calculate the deflection of the beam at all points.

The function that you are to find the root of should return the value of the deflection of the beam at the right endpoint given the value of the slope of the beam at the left endpoint. The method used to solve this problem is called the Shooting Method. The idea is to adjust the slope at the left endpoint until the beam passes through 0 at the right endpoint. It is based on the concept of adjusting the elevation (slope) of a cannon so that a cannon ball fired from the cannon will pass through a specified point down range.

The prototype for the forcing routine  $f$  is

```
void f(float x, int n, float *y, float *yp)
```

where  $x$  is a point along the beam,  $n$  is the number of differential equations (2 in this case),  $y$  is an array to contain the displacement and slope at  $x$ , and  $yp$  is an array of length 2 defined by:

```
yp(0) = y(1)
```

```
yp(1) = -lambda * sin(y(0))
```

where  $\lambda$  is the compression load  $\lambda$  applied to the beam. Your main program should get the value of  $\lambda$ , start the root finder with initial values of 4 then 5 for the slope at the left endpoint, and a tolerance of  $5e-6$ . You should use 0.01 as the distance between points on the beam.

The one step differential equation solver is defined by the following prototype:

```
void rk(float x, float h, int n, float y, void (*f)(float, int, float *, float *))
```

where  $x$ ,  $n$  and  $y$  are defined as above,  $h$  is the distance between successive points along the beam, and  $f$  is the name of the forcing routine described above.