

## Lab 4

Each lab will begin with a recap of last lab and a brief demonstration by the TAs for the core concepts examined in this lab. As such, this document will not serve to tell you everything the TAs will in the demo. It is highly encouraged that you ask questions and take notes.

In order to get credit for the lab, you need to be checked off by the end of lab. For non-zero labs, you can earn a maximum of 3 points for lab work completed outside of lab time, but you must finish the lab before the next lab. For extenuating circumstance, contact your lab TAs and Jennifer Parham-Mocello.

**Pair Programming:** In this lab, you can choose a partner for pair programming. **You must be checked off together. You only need one computer for pair programming.** One person will be the driver, who controls the computer and codes, while the other is the navigator, who observes and advises. After 20 minutes, you will switch driver and navigator, continuing this pattern until the task is complete. Please read more about pair programming and the benefits: [Pair Programming Student Handout](#)

The quiz and linux portion of this lab should be completed independently, but the remainder of the lab can be done using pair programming.

### Quiz (Consent Form) in Class Section of Canvas

1. At this time, you need to consent or not consent to be in the research study regarding your recitation designs. Go to **Student Consent Form** in the class section of Canvas.

### (2 pts) Lab Quiz from Lab #3

2. Vim Commands:
  - How do you auto indent your program?
  - Explain what the following commands do: dd, y3, p, :set cindent

### (1 pt) VIM Exercises

These **exercises on the computer need to be repeated by each student** in the pair. This is to ensure that both students understand how to get around in Linux!!! For this part of the lab, you will create a .vimrc file that will help you develop your C++ programs using Vim. First, we need to create a simple .vimrc file in your home directory on the engr server.

**vim .vimrc**

In this file, you can insert the following lines so that it makes working with vim easier. Comments are prefaced with a quotation mark, “.

```
filetype on
filetype plugin on
filetype indent on
autocmd FileType c, cpp
set cindent “This allows for c-like indentation
set sw=3 “Set the sw for the auto indent to 3 spaces
set number “Show the line numbers on the left
```

```

"Change the color of the text and turn on syntax highlighting
"color desert
color torte
colorscheme evening
syntax on "Turn on syntax highlighting
set showmatch "Show matching braces
set showmode "Show current mode
"When one of the chars is typed, the matching is typed and the cursor moves left
"The line below is single quotes
inoremap ' "<Left>
"The line below is double quotes
inoremap " ""<Left>
inoremap { {}<Left>
inoremap ( ()<Left>

```

There are many more commands you can insert in this file, and here is a reference guide to some of these: <http://vimdoc.sourceforge.net/html/doc/starting.html>

### (5 pts) Quadratic Equation and Area Under the Curve

#### Quadratic Equation – Void Function:

Now, design and implement an additional function to calculate the quadratic equation given the a, b, and c values. Since we cannot return 2 values for the x, then just print each x value. The quadratic equation is as follows.

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

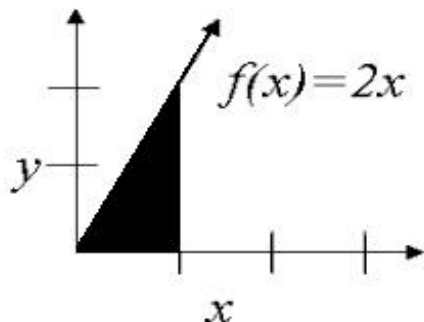
**Ask the user for the a, b, and c values, and output the values for x.**

- What are the types for the a, b, c, and x values?
- What calculations will you use?

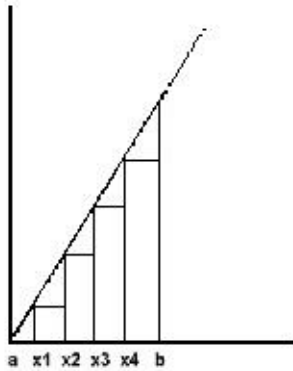
#### Integration – Value-Returning Function:

Now, we are going to find the area under a curve using rectangles. We will use **n** rectangles between two points on the x axis to estimate the area under the curve at this location.

#### Background/Example:



## Rectangle Numerical Method:



$n$  = number of rectangles (user input)  
 $a$  = beginning x value (user input)  
 $b$  = ending x value (user input)  
 $w$  = width of each rectangle,  $(b-a)/n$   
 $f(x)$  = height of rectangle  
Area =  $w \cdot f(x)$

- What are you using for the x value to send to the function,  $f(x)$ , e.g.  $a$ ,  $x_1$ ,  $x_2$ , etc. in the above picture?
  - What is the initial value of  $x$ ?
  - How will you determine the subsequent values?
- What kind of loop are you using? Construct this loop.
  - What are the loop counter values and condition?
  - What statements are in the loop?

You need to create a C++ function called  $f$  to calculate the y value, height, given an x value,  $f(x)$ . You can define this function above main or you can define it below main and include a prototype, `double f(double);`, above main. In addition, you need to fill in the function header.

```
//Name: f
//Description:
//Pre Conditions:
//Post Conditions: none
//Return:

double f(double x) {
    return 2*x;
}
```

Now create a C++ function called  $area$  to calculate the area under the curve given the starting and ending x value and number of rectangles. You can define this function above main or you can define it below main and include a prototype, `double area(double, double, int);`, above main. In addition, you need to fill in the function header.

```
//Name: area
//Description:
//Pre Conditions:
//Post Conditions: none
//Return:
```

```
double area(double a, double b, int n) {  
    //calculate and return the area under the function, f(x)  
  
}
```

**Believe it or not, but some of you have just completed your first integral!!!!** We cannot take integrals (or perform calculus operations) in the computer, so instead we use numerical methods to approximate the solution. You can test your area using the following link:  
<http://www.wolframalpha.com/widgets/view.jsp?id=8ab70731b1553f17c11a3bbc87e0b605>

### **(2 pts) Testing and Debugging**

Download the following cpp file.

<http://classes.engr.oregonstate.edu/eecs/fall2017/cs161-001/labs/buggyCode.cpp>

You must find as many bugs as possible and fix them. Some are logic errors, some are syntax errors. Hint: there are 17 bugs, some obvious, some more complex. **Make sure you re-compile and run your program after you make a single fix to a mistake to make sure you actually fixed the mistake, didn't introduce new errors, and/or eliminated other errors as a result of the fix.**

### **Extended Learning (If you finish early)**

Make sure that the n, a, and b values in the area under the curve program are fail proof. You should catch non-positive integers for n and any signed floating point number for a and b.