Jason Dorweiler
CS165 Assignment 3

Requirements:

Here's a list from each requirement for what I think needs to be done:

Part 2a.
Set up a basic hello world program just to get something working. Then add a get_float() function that will get a string. The function then get a float input from the user and returns it. Make another function that takes in three floats and checks to see if the first argument is in between the other two arguments. Return the first argument if it is and return -1.0 if it isn't. Make a third function that takes in two floats (width, height) and calculates and returns the area of a rectangle.

Part 2b.
Make functions for each of the curves to calculate integrals from.

Part 2c.
Make another function fun_funcs(int, float) that checks to see that the user entered a valid function to calculate. Use a switch statement to do this.

Part 2d.
Testing that everything works.

Part 2f.
Get the code to calculate the integral. Store the area in a total variable. Get the start, stop, and slices from the user. Then get it to actually return the correct area.

Possible solution:

Part 2a.
Pseudo code for the three functions to set up:

```
get_float( string input){
    cout << string input
    cin >> float from user
    return float
}

check_valid_input (check, low, high){
    if ( check < low || check > high){
        return -1.0
    }
    else{
        return check
    }
}

calc_rect_area(width, height){
    return width*height
}
```

Part 2c.

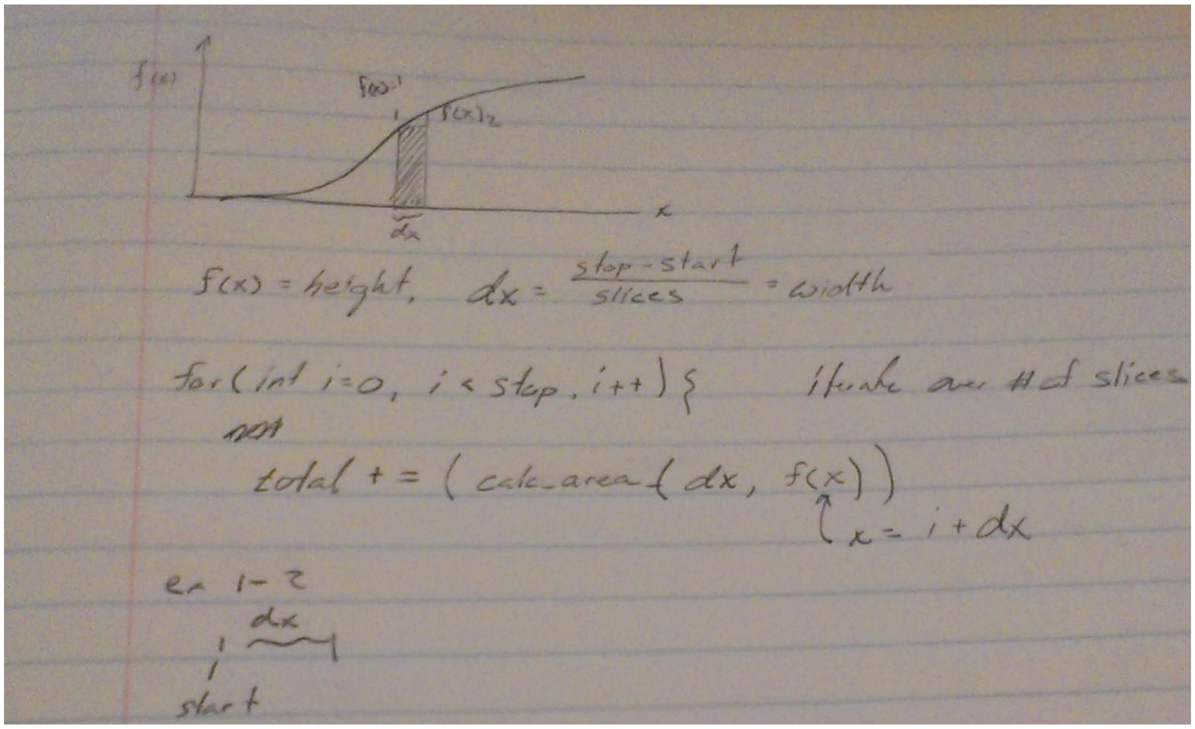
This function will do some error checking on the input to see if the user actually entered a valid function.

```
fun_funcs(function, input){
        // first check to see if it's valid
        while( function is not in between 1 and 5){
                cout << not a valid function
                cin >> func
        }

        // then use a switch to actually do the calculation.  Maybe this can be recursive?  Just have it call the same function
        over again and keep track of the total in a global variable..

        swtich (func)
                case;  each case will then call the correct function
}
```

Here's some hand written notes I used later on to figure out the integral calculation part.


f(x)
f(x)_1
f(x)_2
x
dx

f(x) = height, $dx = \frac{stop - start}{slices}$ = width

for(int i=0, i < stop, i++){   iterate over # of slices

~~[illegible]~~

    total += (calc_area(dx, f(x)))

                $\uparrow$ x = i + dx

ex 1-2

dx

start

Implementation

See comments in the source file.

Testing and Debugging

For testing the code will check to see if the user enters a valid function number (1-5) and that what ever gets entered doesn't cause a cin error. If it does then it will loop again asking to re-enter the input. Then in part of the loop where it asks for the bounds of the integral I check for an input that doesn't throw a cin error.

For debugging I did have several lines where I had to print out the width and height as it was going through each loop. I ended up finding a bug in my code where I incremented through each slice in the integral but I was starting the loop at i=1 where I need to be using i=0. This difference caused it to start one slice ahead of where it should. It was really hard to figure out what was going on. For a large number of slices being off by one slice is only a very tiny error and so my integrals were coming out looking perfect. But for a small number of slices it was completely wrong. The person I partnered with ended up finding this bug too and helped figure out what was going on.

Reflection

This assignment was pretty understandable. I actually use the Riemann sum method at work to calculate gas volumes in reactors from sensor data using Excel. So I'm pretty familiar with how to do this. The bug with the slices was tough to track down but I'm glad someone else looked at and tested my code because I probably wouldn't have noticed it since I was getting the right answer for large slices. It was nice to see someone else's code too. I saw some ways that he had done things differently and found a few things I had missed.

Here's his message to me about my code:

Hey Jason,

Thanks for the notes, and after looking at your code I kinda feel embarrassed about mine... yours is much cleaner and more intuitive. There are few things from your code that make so much more sense than the way I chose to go about it. I definitely need to get better about verifying inputs and you're absolutely right about using the check_valid_input, I tried it a few times but kept getting stuck.

Here are your grades

Ease of use : 3
Readability : 3
Meet Specifications : 2.99 (notes below)

The only slight issue I found and it could just be my calculations, but for smaller parameters the total area seems to be a little off. Quick example, I tried f1, with start of 0 and stop of 2 with 4 slices and I get 26.875 instead of -.125. However when I try larger inputs it gives the correct answer, I've looked through your loop and it all makes sense as to how it should work. I went through it a few times but I can't put my finger on what might be going on, I did try changing some of the loop and came up with what's below (it appears to work with large and small parameters... it's just not nearly as compact as yours). Thanks again and I really appreciate you taking the time to provide really solid notes and feedback.

```
 for(int i = 1; i <= slices; ++i){
     switch(func){
         case 1: height = f1(start);
```

```
            // cout << "i*width" << i*width << "height " << height << endl; debug
             start += width;
             total += calc_rect_area(width, height);
             break;
        case 2: height = f2(start);
             start += width;
             total += calc_rect_area(width, height);
             break;
        case 3: height = f3(start);
             start += width;
             total += calc_rect_area(width, height);
             break;
        case 4: height = f4(start);
             start += width;
             total += calc_rect_area(width, height);
             break;
        case 5: height = f5(start);
             start += width;
             total += calc_rect_area(width, height);
             break;
      }
   }
```

And one small note, line 177 has a small typo : 2x^5 instead of 2x^2

Thanks again!

Here's my grades and comments for his code

Hey Victor,

Looks pretty good and after looking at your code I noticed that I
didn't use the get_float function in mine.

Here's the grades:

Ease of use: 3
Readability: 3
Meets specifications: 2 (see below)

One problem though, your integral calculation is wrong when you enter
a large number of slices. For example try f1, from 1 to 2 with
10000000 slices. It should give you 11.75 back but I'm getting 18.55.
Your calculation seems to work for smaller slices but something is
throwing it off with the larger slices. To fix it change line 175 to:

```
float slice_height = fun_func(func, start+i*slice_width);
```

Remove

start+= slice_width

I'm still not a 100% sure why your way doesn't work though. It seems like it should. Actually when you print out the values for the slices after each loop your way looks good and mine looks wrong because it ends up past the upper limit of the integral. Let me know if you figure it out!

Other small notes:

Line 104 - It's a good idea to check to see that you get an int here using !cin. I also think you should be using the check_valid_input function to do this? Maybe add a while loop here too so that a user can enter in the choice again if they mess it up. Right now it just exits.