ECE 150 Coding Lab 1: Triangles

Submission due *at the end of the Lab Period*

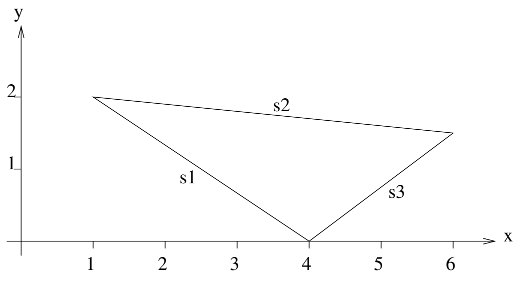
Absque sudore et labore nullum opus perfectum est

Basic math, `floats`, and "if" statements

You are required to write a program, `triangle` that accepts six (6) command-line arguments. These six arguments correspond to three (3) Cartesian points that should form a triangle. The basic requirement is to calculate the area of this triangle. If the program is executed as:

```
        ./triangle 1 2 4 0 6 1.7
```

then it is to compute the area of this triangle:



You can calculate the area, a, of a triangle, given three points, using Heron's formula:

$$s = \frac{1}{2}(s_1 + s_2 + s_3)$$

$$a = \sqrt{s(s - s_1)(s - s_s)(s - s_3)}$$

where s_1, s_2, and s_3 are the lengths of the three sides of the triangle.[1] The output should be:

```
The area of the triangle formed by points <p1>, <p2>, and <p3> is: <a>
```

where `<p1>`, `<p2>`, and `<p3>` are the three Cartesian points and `<a>` is the area. In the above example, the output should be:

```
The area of the triangle formed by points (1,2), (4,0), and (6,1.7) is: 4.55
```

Submit your work to http://marmoset01.shoshin.uwaterloo.ca project **L1-Triangle**. Your submission file must be named `Triangle.cpp`. Note that the filename is *case sensitive*.

Knowledge required to complete this exercise:

- Access to `argv[]` items
- Use of `atof()` and `sqrt()` functions
- `float` variables and basic arithmetic operators
- Use of `cout`, `endl` and `std`

[1] See https://www.wikihow.com/Calculate-the-Area-of-a-Triangle Method 2 for a detailed explanation.

ECE 150 HOMEWORK 1: GETTING STARTED: MATH PROBLEMS

SUBMISSION DUE WEDNESDAY, SEPTEMBER 12th AT 10:00 PM

Pithy Quote #47

1. Robust Triangles [1 marks]

The in-lab exercise only required that you compute the area of a triangle given three Cartesian points. However, the three points may not form a triangle. Two, or even all three, points may be identical, in which case the points either form a single point or a straight line. Worse, there may not be six command-line arguments; there could be fewer, in which case your program cannot perform the calcuation and should issue an error message instead and exit, or too many, in which case you should at least inform the user that some extra command-line arguments are being ignored.

- Check that there are six arguments. If there are fewer than six arguments, you should output the error message (to `cerr`):

  ```
  Error: Expected 6 arguments; received <n>; exiting
  ```

  where `<n>` is the number of arguments actually received. You should return `-1` in your return statement from `main()`. If there are more than six arguments, you should output the warning message (to `cerr`):

  ```
  Warning: Expected 6 arguments; received <n>; ignoring extraneous arguments
  ```

  where `<n>` is the number of arguments actually received.
- Check that they are not three overlapping points
- Check that it is not a line (if it is, what is the slope?)

Submit your work to http://marmoset01.shoshin.uwaterloo.ca project **H1-RobustTriangle**. Your submission file must be named `RobustTriangle.cpp`. Note that the filename is *case sensitive*.

Additional knowledge required to complete this exercise:

- `argc`
- `if()` and `else`
- Approximation comparison of `float` values
- Use of `cerr` for warnings and error messages
- `return` value from `main()`

2. Triangle Type [1 marks] Identify the type of triangle. Specifically, a triangle may be equilateral, isosceles, or scalene. Further, it can be acute, right, or obtuse.[2]

The output should be:

```
The triangle formed by points  <p1>, <p2>, and <p3> is: <type of triangle>
```

where `<p1>`, `<p2>`, and `<p3>` are the three Cartesian points and `<type of triangle>` is the type of the triangle. In the example from the in-lab exercise, the output should be:

```
The triangle formed by points (1,2), (4,0), and (6,1.7) is: scalene obtuse
```

Submit your work to http://marmoset01.shoshin.uwaterloo.ca project **H1-TriangleType**. Your submission file must be named `TriangleType.cpp`. Note that the filename is *case sensitive*.

Additional knowledge required to complete this exercise:

- `else`
- `&&`
- `||`

[2]It is assumed that you either know, can recall, or can work out how to determine the type of a triangle given the points that form the triangle.

3. Square Root Approximation [1 marks]

How does a computer determine the square root of a number? Answer: successive approximation. The Babylonian method[3] for computing the square root of S (also called Heron's method, named for Heron of Alexandria) is to start with an initial estimate, x_0, and compute a better estimate based on dividing it into the number S and averaging that with the estimate:

$$x_{i+1} = \frac{x_i + \frac{S}{x_i}}{2}$$

where x_i is the i^{th} estimate of the square root of S and x_{i+1} is the next (*i.e.,* the $(i+1)^{th}$) estimate. This method works because if $x_i < \sqrt{S}$ then $\frac{S}{x_i} > \sqrt{S}$ and if $x_i > \sqrt{S}$ then $\frac{S}{x_i} > \sqrt{S}$, which means that in either case the average of those two will be a better estimate than x_i. We stop iterating when the result is *sufficiently* precise. If we want a precision of $p = 0.001\%$[4] then we stop iterating when:

$$\left| \frac{S - (x_{i+1}^2}{S} \right| \leq \text{p}$$

For this question, you are required to write a program that will compute (without using `<math.h>`, *etc.*) the square root of non-negative floating-point numbers using the method described above.

The program will be executed as follows:

```
./SquareRoot <number>
```

where "`<number>`" is a floating-point number whose square root your program is computing. The output should be of the form:

```
The square root of <number> is ...
```

The precision you output should be the actual accuracy of your result, per the above definition, not the requested precision. For example, if invoked as:

```
./SquareRoot 150
```

will result in (in my implementation):

```
The square root of 150 is 12.2474
```

Submit your work to http://marmoset01.shoshin.uwaterloo.ca project **H1-SquareRoot**. Your submission file must be named `SquareRoot.cpp`. Note that the filename is *case sensitive*.

Additional knowledge required to complete exercise:

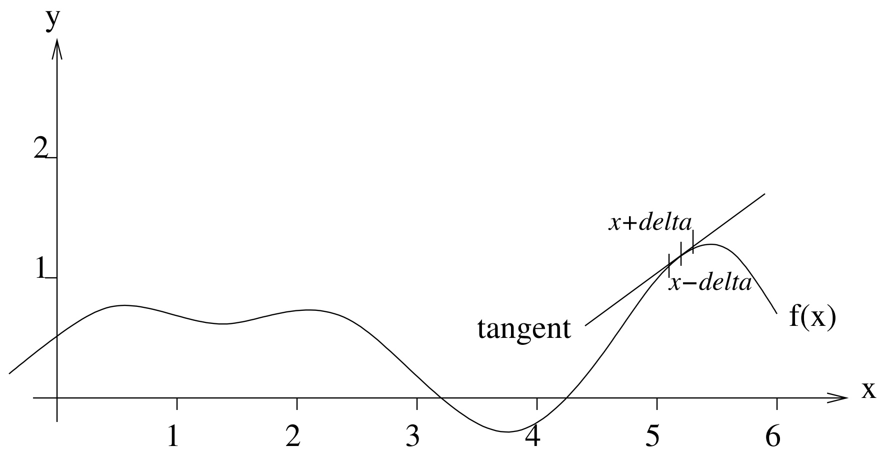
- `while()`

[3]Yes, computing the square root is that old. The Yale Babylonian Collection, which is over 3,500 years old, has the square root of 2 with an accuracy that demonstrates it must have been calculated, not measured; the Rhind Mathematical Papyrus from Egypt, also over 3,500 years old, shows an Egyptian method for computing square roots.

[4]Note the "%"; this means $p = 0.00001$; don't stop early.

4. Differentiation (Instantaneous slope of a function) [1 marks]

Consider the function $f(x)$ illustrated as:



Computing the derivative of a function requires (a) having the function in symbolic form (*e.g.*, $f(t) = 3t\sin(t) + t^2\tan(t) + \ln(t)$) (b) being able to compute the symbolic derivative of that function. It is often the case that there is no symbolic form of a function, but rather just a series of data points, with interpolation used to fill in the gaps between those points. For example, the velocity of a car can be sampled, in which case we would have a function $v(t)$ that is represented by these sample points. The derivative of $v(t)$ is the acceleration of the car.

For this problem, you are given a function declaration:

```
extern float f(float t);
```

that defines some function $f(t)$. You are required to write a program **Derivative.cpp** that takes as input a single value and computes the approximate slope of the function $f(t)$ at that value. The program is invoked as:

```
./Derivative <t>
```

where `<t>` is the value where you wish to compute the slope. The output should be:

```
The slope of f(t) at t = <t> is ...
```

The way you estimate the slope is to compute the value of the function at $t - \delta$ and at $t + \delta$ and then use the slope calculation we covered in class. For the purpose of this assignment, you may assume $\delta = 0.0001$.

Submit your work to http://marmoset01.shoshin.uwaterloo.ca project **H1-Derivative**. Your submission file must be named `Derivative.cpp`. Note that the filename is *case sensitive*.

Additional knowledge required to complete exercise:

- Declaration and use of a function