Data Structures and Objects
CSIS 3700

Fall Semester 2017 — CRN 42034

Project 1 — Fractions
Due date: Friday, September 15, 2017

Goal

Develop and implement a **`Fraction`** class and use it to calculate the intersection point of two line segments.

Details

▹*The `Fraction` class*

The first step in this project is to create a **`Fraction`** class that creates **`Fraction`** objects and performs basic arithmetic on fractions. A **`Fraction`** object contains an integer numerator and denominator with two properties:

- The numerator and denominator have no common factor other than 1;
- The numerator is always nonnegative.

A **`Fraction`** object must support the following actions:

- Initialize a fraction
  - This can initialize to a default value (`0/1`), provide just a numerator (*n*`/1`) or provide both a numerator and denominator
- Set a fraction to a specific value
  - This has the same options as initialize
- Set one fraction equal to another
  - This must be done with the `=` operator
- Add two fractions
  - **`Fraction operator+(const Fraction &a)`**
- Subtract two fractions
- Multiply two fractions
- Divide two fractions
- Compare two fractions — all six comparisons
- Compare a fraction to an integer — all six comparisons
- Input and output fractions
  - **`friend ostream& operator«(ostream &,const Fraction &)`**

  - `friend istream& operator»(istream &,Fraction &)`

The arithmetic functions must use the arithmetic operators `+`, `-`, `*` and `/`.

▹*Source code layout*

The project is also an exercise in *separate compilation*, where the source code is divided into multiple parts. The project consists of four files:

- A header file that contains the class definition;
- A file containing the class methods (functions);
- A file containing a `main` function that performs the intersection calculations
- A *Makefile* that directs the process of creating the executable file

▹*Calculating intersections*

Note: This is taken from https://stackoverflow.com/questions/563198/how-do-you-detect-where-two-line-segments-intersect

Also note: In the following calculations, bold names are points and other names are scalar values.

Given two line segments $\overline{\mathbf{p_1 p_2}}$ and $\overline{\mathbf{q_1 q_2}}$, first compute:

- $\mathbf{r} \leftarrow \mathbf{p_2} - \mathbf{p_1}$
- $\mathbf{s} \leftarrow \mathbf{q_2} - \mathbf{q_1}$

This parameterizes the line segments, so that any point on $\overline{\mathbf{p_1 p_2}}$ can be written as $\mathbf{p} = \mathbf{p_1} + t \cdot \mathbf{r}$ where $0 \le t \le 1$ and any point on $\overline{\mathbf{q_1 q_2}}$ can be written as $\mathbf{q} = \mathbf{q_1} + u \cdot \mathbf{s}$ where $0 \le u \le 1$. The intersection problem can be recasted to solve for t and u.

Next, let $\mathbf{z} \leftarrow \mathbf{q_1} - \mathbf{p_1}$ and calculate the following values:

- $x_t \leftarrow \mathbf{z} \times \mathbf{s}$
- $x_u \leftarrow \mathbf{z} \times \mathbf{r}$
- $y \leftarrow \mathbf{r} \times \mathbf{s}$

The cross product of two points is given by $\mathbf{a} \times \mathbf{b} = \mathbf{a_x} \cdot \mathbf{b_y} - \mathbf{a_y} \cdot \mathbf{b_x}$. Note that it is a scalar (single) value in our context.

The intersection is now given in the following scenarios:

- If $y = 0$ and $x_t = 0$, then the line segments are collinear.
- If $y = 0$ and $x_t \neq 0$, then the line segments are parallel.
- If $y \neq 0$, then calculate $t \leftarrow x_t / y$ and $u \leftarrow x_u / y$.
  - If $0 \le t \le 1$ and $0 \le u \le 1$, then the line segments intersect at $\mathbf{p_1} + t \cdot \mathbf{r}$.
  - Otherwise, the line segments do not intersect.

Your program must determine which case applies and output an appropriate message. If the segments intersect, output the intersection point.

Extra Credit

For 10% extra credit, write a **`Point`** class that implements the following:

- Input and output of a point in the form (x, y) where x and y are two **`Fraction`** objects
- Add and subtract two points using **`operator+`** and **`operator-`**
- Multiply two points, computing the cross product
- Multiply a point by a **`Fraction`**, which multiplies both of the point's coordinates by the given fraction.

Note that both forms of multiply should use **`operator*`**; for cross product the parameter should be a **`Point`** and for scaling it should be a **`Fraction`**.

Note that you must use your **`Point`** class in the project's solution in order to receive the extra credit.

What to turn in

Turn in your source code and **`Makefile`**.