Exercises for the Lecture Series

"Object-Oriented Programming for Scientific Computing"

Dr. Ole Klein
ole.klein@iwr.uni-heidelberg.de

To be handed in on 02. 05. 2017 before the lecture

This is the first exercise sheet for the lecture. Please note that, in contrast to later sheets, you have approximately two weeks to work on the exercises. This is meant as an opportunity to accustom yourself with the exercises and potentially find exercise partners. You should also use this time to get acquainted with Git (at least the basic git pull / add / commit / push cycle) and the web interface of GitLab if these tools are new for you.

Please form groups of three to four people for the exercises. Details about correction and grading depend upon the availability of tutors and the number of submissions. There will only be a short meeting on 20.4. and no exercise group, because things are still being set up, but you are welcome to attend if you have questions about the lecture or exercises or something else to discuss.

EXERCISE 1 RATIONAL NUMBERS

Write a class for rational numbers. The number should always be represented as a *fully reduced fraction* of the form

$$\frac{\text{numerator}}{\text{denominator}}$$

with denominator > 0.

1. What is an appropriate data structure for rational numbers?
2. Start by writing a function `int gcd(int,int)` (greatest common divisor), you will need it to reduce fractions.
   - You can use the Euclidean algorithm to determine the greatest common divisor.
   - For an algorithm see `http://en.wikipedia.org/wiki/Greatest_common_divisor`
   - Implement this scheme as a recursive function.
3. Write a class `Rational`, which represents a rational number. The constructor should have the numerator and the denominator as arguments. Be sure to check for valid input. In addition, the class has two functions `numerator()` and `denominator()` that return the values of the numerator and denominator. The class should have three constructors:
   - a default constructor that initializes the fraction with 1,
   - a constructor that initializes the fraction with a given numerator and denominator and
   - a constructor that initializes the fraction with a given whole number.
4. Supplement the class with operators for `*= += -= /=` and ==.
5. Use the newly implemented methods to implement free operators `* + - /`.
6. Check your implementation using various test cases. Initialize three fractions

$$f_1 = -\frac{3}{12}, \quad f_2 = \frac{4}{3}, \quad f_3 = \frac{0}{1}.$$

Test the operators with the following examples:

$$f_3 = f_1 + f_2, \quad f_3 = f_1 \cdot f_2, \quad f_3 = 4 + f_2, \quad f_3 = f_2 + 5, \quad f_3 = 12 \cdot f_1, \quad f_3 = f_1 \cdot 6, \quad f_3 = \frac{f_1}{f_2}.$$

Print the result after each operation. The corresponding solutions are:

$$\frac{13}{12}, \quad -\frac{1}{3}, \quad \frac{16}{3}, \quad \frac{19}{3}, \quad -\frac{3}{1}, \quad -\frac{3}{2}, \quad -\frac{3}{16}.$$

10 Points

EXERCISE 2 FAREY SEQUENCES

A Farey sequence F_N of degree N (Farey fractions of degree N) is an ordered set of reduced fractions

$$\frac{p_i}{q_i} \qquad \text{with} \qquad p_i \le q_i \le N \qquad \text{and} \qquad 0 \le i < |F_N|$$

and

$$\frac{p_i}{q_i} < \frac{p_j}{q_j} \qquad \forall\, 0 \le i < j < |F_N|.$$

Use the class `Rational` from the first exercise to write a function

```
void Farey(int N)
```

which calculates the Farey fractions up to degree N and prints the resulting Farey sequences up to degree N on the screen.

Algorithm: The sequences can be computed recursively. The first sequence is given by

$$F_1 = \left(\tfrac{0}{1}, \tfrac{1}{1}\right)$$

For a known sequence F_N one can get F_{N+1} by inserting an additional fraction $\frac{p_i+p_{i+1}}{q_i+q_{i+1}}$ between two consecutive entries $\frac{p_i}{q_i}$ and $\frac{p_{i+1}}{q_{i+1}}$ if $q_i + q_{i+1} = N + 1$ holds for the sum of denominators.

Example: Determining F_7 from F_6 results in the following construction:

$$F_6 = \Big(\underbrace{\tfrac{0}{1}, \tfrac{1}{6}}_{\frac{1}{7}}, \tfrac{1}{5}, \underbrace{\tfrac{1}{4}, \tfrac{1}{3}}_{\frac{2}{7}}, \underbrace{\tfrac{2}{5}, \tfrac{1}{2}, \tfrac{3}{5}}_{\frac{3}{7} \text{ and } \frac{4}{7}}, \underbrace{\tfrac{2}{3}, \tfrac{3}{4}}_{\frac{5}{7}}, \tfrac{4}{5}, \underbrace{\tfrac{5}{6}, \tfrac{1}{1}}_{\frac{6}{7}}\Big)$$

The new elements are:

$$\tfrac{0+1}{1+6} = \tfrac{1}{7} \;;\; \tfrac{1+1}{4+3} = \tfrac{2}{7} \;;\; \tfrac{2+1}{5+2} = \tfrac{3}{7} \;;\; \tfrac{1+3}{2+5} = \tfrac{4}{7} \;;\; \tfrac{2+3}{3+4} = \tfrac{5}{7} \;;\; \tfrac{5+1}{6+1} = \tfrac{6}{7}$$

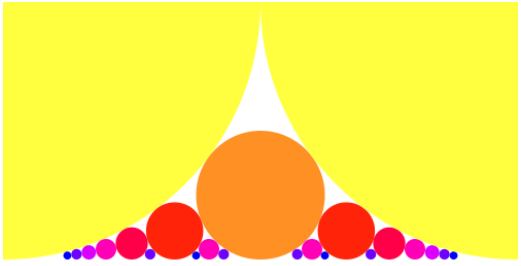
The sorted sequence then is:

$$F_7 = \left(\tfrac{0}{1}, \mathbf{\tfrac{1}{7}}, \tfrac{1}{6}, \tfrac{1}{5}, \tfrac{1}{4}, \mathbf{\tfrac{2}{7}}, \tfrac{1}{3}, \tfrac{2}{5}, \mathbf{\tfrac{3}{7}}, \tfrac{1}{2}, \mathbf{\tfrac{4}{7}}, \tfrac{3}{5}, \tfrac{2}{3}, \mathbf{\tfrac{5}{7}}\ \tfrac{3}{4}, \tfrac{4}{5}, \tfrac{5}{6}, \mathbf{\tfrac{6}{7}}, \tfrac{1}{1}\right)$$

For checking:
The Farey sequences up to degree 6

$$\begin{aligned}
F_1 &= \left(\tfrac{0}{1}, \tfrac{1}{1}\right) \\
F_2 &= \left(\tfrac{0}{1}, \tfrac{1}{2}, \tfrac{1}{1}\right) \\
F_3 &= \left(\tfrac{0}{1}, \tfrac{1}{3}, \tfrac{1}{2}, \tfrac{2}{3}, \tfrac{1}{1}\right) \\
F_4 &= \left(\tfrac{0}{1}, \tfrac{1}{4}, \tfrac{1}{3}, \tfrac{1}{2}, \tfrac{2}{3}, \tfrac{3}{4}, \tfrac{1}{1}\right) \\
F_5 &= \left(\tfrac{0}{1}, \tfrac{1}{5}, \tfrac{1}{4}, \tfrac{1}{3}, \tfrac{2}{5}, \tfrac{1}{2}, \tfrac{3}{5}, \tfrac{2}{3}, \tfrac{3}{4}, \tfrac{4}{5}, \tfrac{1}{1}\right) \\
F_6 &= \left(\tfrac{0}{1}, \tfrac{1}{6}, \tfrac{1}{5}, \tfrac{1}{4}, \tfrac{1}{3}, \tfrac{2}{5}, \tfrac{1}{2}, \tfrac{3}{5}, \tfrac{2}{3}, \tfrac{3}{4}, \tfrac{4}{5}, \tfrac{5}{6}, \tfrac{1}{1}\right).
\end{aligned}$$

There is a beautiful illustration of these fractions, the Ford circles [a]:



[a] see `http://en.wikipedia.org/wiki/Ford_circle`

10 Points