CMPUT 350 Lab 4 Exercise Problems

Rules:

- You can use all course material and man pages, but no other information such as web pages, books, or written notes. Using other information sources during the exercise constitues cheating.
- Your programs must compile without warning using

  ```
  g++ -Wall -Wextra -Wconversion -Wsign-conversion -O -g -std=c++17 ...
  ```

  In case there are compiler warnings or errors, marks will be deducted.
- Test your programs with different values. For now, the speed of your program is irrelevant. So don't spend time on optimization
- You must check for the appropriate preconditions/postconditions. Your program shouldn't crash or have undefined behaviour (**hint**: use asserts)!
- Your programs must be well structured and documented. Use ctrl-x t in Emacs to pretty-print it. Marks are assigned to functionality, program appearance, and comments.
- In case your program hangs, use ctrl-c to terminate it.
- Remember that you need to include the appropriate header files. To find out which ones you need for specific functions such as printf, use then man command.

Submit your solution files `Rational.h Rational.cpp main.cpp` on eClass under "Lab 4 / Submission".

Important: Submit often (the system will only accept solutions submitted before 16:50)

---

1. [36 marks] In this exercise you will implement a rational number type that is meant as an exact arithmetic replacement for rounding-error-prone floating point types. Rational numbers are represented as two integer variables: numerator `num` and denominator `den > 0`. The rational number they represent is num/den.

```
1 class Rational {
2 public:
3     ...
4 private:
5     // only data
6     int num, den;
7 };
```

At all times, `den` must be `> 0`. In this exercise we won't deal with arithmetic overflow or simplifying rational numbers by dividing num,den by their greatest common divisor. However, division by 0 must be checked with an assertion, and `operator==` must work correctly, i.e. two rational numbers are equal if the ratios represent the same value (e.g., `2/4 == 4/8`).

Declare operators and methods in `Rational.h` and implement all of them in `Rational.cpp` so that the given code in `main.cpp` works.

ALL operators and methods have to be implemented in `Rational.cpp` File `main.cpp` must contain all test code AND code that triggers division by 0 and an illegal constructor call (division by 0), **but you may leave that code commented out so that your code runs without crashing but that we can see that you have tested your code for those cases.**