La práctica consiste en desarrollar algoritmos para diversas **operaciones con números racionales (fracciones)**, y algunas de ellas usando comparación de números reales.

Guión

Se dispone del siguiente material:

- Un fichero **`rational_t.hpp`** con la definición de la clase número racional.
- Un fichero **`rational_t.cpp`** con la implementación de dicha clase número racional.
- Un fichero **`main_rational_t.cpp`** con ejemplos de uso de la clase.

Desarrollo

FASE I. Comprensión y comentario del material proporcionado

- Descargar los ficheros fuente.
- Compilarlos y ejecutarlos:

```
$ g++ -g rational_t.cpp main_rational_t.cpp -o main_rational_t
$ ./main_rational_t
```

- Examinar el **fichero de cabecera (.hpp) identificando y comentando adecuadamente** los constructores, el destructor, los métodos para acceder a los atributos, los métodos para lectura desde teclado y escritura a pantalla, e identificar los elementos privados y públicos, tanto atributos como métodos.
- Examinar el **fichero de implementación (.cpp) para comprobar y comentar** el código de todos los métodos definidos en la clase.
- Identificar las **pautas de estilo de programación de C++** que se comentan en el código fuente según las directrices dadas en "C++ Programming Style Guidelines".
- **Rellenar los datos** de `AUTOR`, `FECHA`, `EMAIL`, etc. que aparecen al principio de cada fichero fuente.

FASE II. Desarrollo de métodos de comparación

Dados dos números reales $\mathbf{a, b} \in \mathbf{R}$, y una tolerancia o precisión $\boldsymbol{\epsilon}$,

- para comprobar que a y b son iguales se tiene que verificar que $\mathbf{|a-b|<\epsilon}$ $\mathbf{|a-b|<\epsilon}$, es decir, que el valor absoluto de la diferencia es menor que cierta precisión;
- para comprobar que **a** es mayor que **b** se tiene que verificar que $\mathbf{a-b>\epsilon}$;
- para comprobar que **a** es menor que **b** se tiene que verificar que $\mathbf{a-b<-\epsilon \Longrightarrow b-a>\epsilon}$;
- para comprobar que **a** es igual a cero se tiene que verificar que $\mathbf{|a|<\epsilon}$.

Se deben desarrollar tres funciones booleanas:

```
bool is_equal(const rational_t& r, const double precision = EPSILON)
const;
bool is_greater(const rational_t& r, const double precision = EPSILON)
const;
bool is_less(const rational_t& r, const double precision = EPSILON)
const;
```

como métodos de la clase **`rational_t`**, que efectúen la comparación de los valores reales de los números racionales tal y como se ha descrito.
Nota: Se deberá hacer uso de la función **`fabs(double a)`**, para lo cual se deberá cargar la cabecera **`#include <cmath>`**.

FASE III. Desarrollo de las operaciones aritméticas sobre números racionales

Se deben desarrollar los métodos:

```
rational_t add(const rational_t&);
rational_t substract(const rational_t&);
rational_t multiply(const rational_t&);
rational_t divide(const rational_t&);
```

que implementan las funciones
de **sumar** (*add*), **restar** (*substract*), **multiplicar** (*multiply*) y **dividir** (*divide*) números racionales, respectivamente.

Para dichas operaciones, se puede consultar el siguiente enlace donde se describen de forma simplificada.

Asimismo, dentro del código del procedimiento **`main()`** del
fichero **`main_rational_t.cpp`** se han puesto algunos posibles ejemplos de operaciones.

Evaluación

El criterio de evaluación será el siguiente:

- Concluir Fase I, y haber resuelto satisfactoriamente la modificación propuesta: **5**
- Concluir Fase I, II, y haber resuelto satisfactoriamente la modificación propuesta: **7**
- Concluir Fase I, II, III, y haber resuelto satisfactoriamente la modificación propuesta: **10**