Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für theoretische Informatik
Dr. J. Giesen

12. Januar 2005

Informatik I für D-ITET — Serie 11 — WS 04/05

URL: `http://www.ti.inf.ethz.ch/ew/courses/inf1-ITET/`

Aufgabe 1 [Schnellübung – 20 Min.] (5 Punkte)

Aufgabe 2 (4 Punkte)

Betrachten Sie folgendes Programm-Fragment. Geben Sie für jeden Funktionsaufruf an, welche Funktion jeweils aufgerufen wird und begründen Sie Ihre Antwort.

```
void foo(double, double)      { ... }  // Funktion A
void foo(unsigned int, int)   { ... }  // Funktion B
void foo(float, unsigned int) { ... }  // Funktion C
```

a) `foo(1, 1)`

b) `foo(1u, 1.0f)`

c) `foo(1.0, 1)`

d) `foo(1, 1u)`

e) `foo(1, 1.0f)`

f) `foo(1.0f, 1.0)`

Aufgabe 3 (3 Punkte)

Beweisen Sie, dass der Quicksort Algorithmus aus Vorlesung 10 bei Eingabe von n Elementen höchstens $\binom{n}{2}$ Vergleiche durchführt.

Abgabe: Aufgaben 2+3: am 17. Januar 2005, am Anfang der Uebungen, schriftlich.

Eidgenössische Technische Hochschule Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für theoretische Informatik
Dr. J. Giesen

12. Januar 2005

Informatik I: Material aus der Vorlesung

Programm: rational.C

```
// Programm: rational.C
// Brueche ganzer Zahlen.

// Achtung: Programm compiliert so nicht!

namespace ifet {

  // Klasse zur Repraesentation von Bruechen ganzer Zahlen.
  class Rational {
  public:

    typedef int NT;

    Rational(const NT& n, const NT& d);
    // PRE: d != 0.
    // POST: Bruch initialisiert als n / d.

    Rational(const NT& n);
    // POST: Bruch initialisiert als n / 1.

    const NT& n() const;
    // POST: Rueckgabewert ist Zaehler des Bruchs.

    const NT& d() const;
    // POST: Rueckgabewert ist Nenner des Bruchs.

    void normalize();
    // POST: Zaehler und Nenner sind teilerfremd.

    Rational& operator+=(const Rational& x);
    // POST: x wurde zu *this addiert.

    Rational& operator-=(const Rational& x);
    // POST: x wurde von *this subtrahiert.

  private:

    NT n_; // nominator (Zaehler)
    NT d_; // denominator (Nenner) Invariante: d_ > 0.
  };

  Rational operator+(const Rational& x, const Rational& y)
  // POST: Rueckgabewert ist x + y.
  {
    Rational z = x;
    z += y;
    return z;
  }

  Rational operator-(const Rational& x, const Rational& y)
  // POST: Rueckgabewert ist x - y.
  {
    Rational z = x;
    z -= y;
    return z;
  }

  // ...

} // namespace ifet

int main()
{
  ifet::Rational q(1, 2);
  ifet::Rational r(2, 3);
  ifet::Rational x = q + r;
  ifet::Rational y = q - r;

  std::cout << "q   = " << q.n() << "/" << q.d() << "\n"
            << "r   = " << r.n() << "/" << r.d() << "\n"
            << "q+r = " << x.n() << "/" << x.d() << "\n"
            << "q-r = " << y.n() << "/" << y.d() << std::endl;
  return 0;
}
```