Esercizio

Definire, separando interfaccia ed implementazione, una classe `Raz` i cui oggetti rappresentano un numero razionale $\frac{num}{den}$ (naturalmente, i numeri razionali hanno sempre un denominatore diverso da 0). La classe deve includere:

1. opportuni costruttori;
2. un metodo `Raz inverso()` con il seguente comportamento: se l'oggetto di invocazione rappresenta $\frac{n}{m}$ allora `inverso` ritorna un oggetto che rappresenta $\frac{m}{n}$;
3. un operatore esplicito di conversione al tipo primitivo `double`;
4. l'overloading degli operatori di somma e moltiplicazione;
5. l'overloading dell'operatore di incremento postfisso che, naturalmente, dovrà incrementare di 1 il razionale di invocazione;
6. l'overloading dell'operatore di uguaglianza;
7. l'overloading dell'operatore di output su `ostream`;
8. un metodo `Raz unTerzo()` che ritorna il razionale 0.3333...

```
#include <iostream>
#include <cmath>

class Raz {
private:
  int num, den; // INV globale: den!=0, numero razionale num/den

  // riduzione naive
  void reduce() {
    if(num%den==0){ num=num/den; den=1;}
    else if(den%num==0){den=den/num; num=1;}
    else {
      int i;
      if(abs(num)<abs(den)) i=num/2;
      else i=den/2;
      do {
        if(num%i==0 && den%i==0){num=num/i; den=den/i;}
        i--; // naive
      } while(i>=2);
    }
  }

public:
  // conversione int => Raz bloccata da explicit
  explicit Raz(int n=0, int d=1): num(d==0 ? 0 : n), den(d==0 ? 1 : d) {}

  operator double() const {
    return static_cast<double>(num)/static_cast<double>(den);
  }

  Raz inverso() const {
    return Raz(den,num); // inverso di Raz(0,1) e' Raz(0,1)
  }

  Raz operator+(const Raz& r) const {
    return Raz(num*r.den + den*r.num,den*r.den);
  }

  Raz operator*(const Raz& r) const {
    return Raz(num*r.num,den*r.den);
  }

  Raz operator++(int) { // incremento postfisso
    Raz loc(*this);
    num += den;
    return loc;
  }

  Raz& operator++()  {  // incremento prefisso
    num += den;
    return *this;
  }

  bool operator==(const Raz& r) const {
    return num*r.den == r.num*den;
  }

  static Raz unTerzo() {
    return Raz(1,3);
  }
};

std::ostream& operator<<(std::ostream& os, const Raz& r) {
  return os << "il razionale come double è: " << r.operator double();
}
```

```
int main() {
  Raz x(2,3), y(2), z, u(1,8), v(3,2), w(8,4);

  std::cout << x+y+v*u << std::endl; // 2.85417
  std::cout << u.inverso() << std::endl; // 8
  std::cout << (y == w) << std::endl; // true
  std::cout << y++ << " " << ++w << std::endl; // 2 3
  std::cout << (++y + Raz::unTerzo()) << std::endl; // 4.33333
  std::cout << 2 + Raz(1,2) << std::endl; // 2.5
}
```