Esercizi di Programmazione ad Oggetti

Lista n. 1

Esercizio 1

Perché il seguente programma non compila? Modificare o eliminare una e soltanto una delle righe 1-8 in modo che il programmi compili.

```
class C {
public:                        // 1
  int *const p;                // 2
  C(int a=0): p(new int(a))    // 3
  { }                          // 4
};

main() {
  C x(3);                      // 5
  C y;                         // 6
  x=y;                         // 7
  C z(y);                      // 8
}
```

Esercizio 2

Si consideri il seguente programma.

```
#include<iostream>
using std::cout;

class C {
public:
  int x;
  C(int k=5): x(k) {};
  C* m(C& c) {
    if((c.x != 5) && (x==5)) return &c;
    return this;
  }
};

main() {
  C a, b(2), c(a);
  cout << (b.m(b))->x << ' ' << (a.m(a))->x << ' ' << (b.m(c))->x
       << ' ' << c.m(a) << ' ' << c.m(c);
}
```

Il seguente programma compila correttamente? Se sì, la sua esecuzione quali stampe provoca in output?

Esercizio 3

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...