Programmazione 2
Prima Prova Parziale – 27/10/2003

Nome..... Cognome.....

Matricola..... Laurea in.....

Non si possono consultare appunti e libri. Ogni quesito a risposta multipla ha ESATTAMENTE una risposta corretta: segnare con una croce la risposta scelta. Le risposte errate vengono penalizzate. Dove previsto scrivere CHIARAMENTE la risposta nell'apposito spazio. ATTENZIONE: In tutti gli esercizi si intende la compilazione standard g++ con il flag `-fno-elide-constructors`.

1. Il seguente programma compila.

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

class S {
public:
  string s;
  S(string t): s(t) {}
};
class N {
private:
  S x;
public:
  N* next;
  N(S t, N* p): x(t), next(p) {cout << "N2 ";}
  ~N() {if (next) delete next; cout << x.s + "~N ";}
};
class C {
  N* pointer;
public:
  C(): pointer(0) {}
  ~C() {delete pointer; cout << "~C ";}
  void F(string t1, string t2 = "pippo") {
    pointer = new N(S(t1),pointer); pointer = new N(t2,pointer);
  }
};
main(){
  C* p = new C; cout << "UNO\n";
  p->F("pluto","paperino"); p->F("topolino"); cout <<"DUE\n";
  delete p; cout <<"TRE\n";
}
```

Quali stampe produce la sua esecuzione? Se una istruzione non produce alcuna stampa si scriva **NESSUNA STAMPA**. Si usi l'ultima riga per eventuali stampe che dovessero seguire "TRE".

..... UNO

..... DUE

..... TRE

.....

2. Il seguente programma compila.

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

class A {
private:
  int x;
public:
  A(int k = 5): x(k) {cout << k <<  " A01 ";}
  A(const A& a): x(a.x) {cout << "Ac ";}
  A g() const {return *this;}
};

class B {
private:
  A ar[2];
  static A a;
public:
  B() {ar[1] = A(7); cout << "B0 ";}
  B(const B& b) {cout << "Bc ";}
};
A B::a = A(9);

A Fun(A* p, const A& a, B b) {
  *p = a;
  a.g();
  return *p;
};

main() {
  A a1; cout << "UNO\n";
  A a2(3); cout << "DUE\n";
  A* p = &a1; cout << "TRE\n";
  B b; cout << "QUATTRO\n";
  a1 = Fun(p,a2,b); cout << "CINQUE\n";
  A a3 = Fun(&a1,*p,b); cout << "SEI";
}
```

Quali stampe produce la sua esecuzione? Se una istruzione non produce alcuna stampa si scriva **NESSUNA STAMPA**.

..... UNO

..... DUE

..... TRE

..... QUATTRO

..... CINQUE

..... SEI

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:

   (a) opportuni costruttori;

   (b) un metodo `Raz inverso()` con il seguente comportamento: se l'oggetto di invocazione rappresenta $\frac{n}{m}$ allora `inverso` ritorna un oggetto che rappresenta $\frac{m}{n}$;

   (c) un operatore esplicito di conversione al tipo primitivo `double`;

   (d) l'overloading come metodi interni degli operatori di somma e moltiplicazione;

   (e) l'overloading come metodo interno dell'operatore di incremento postfisso, che, naturalmente, dovrà incrementare di 1 il razionale di invocazione;

   (f) l'overloading dell'operatore di output su `ostream`;

   (g) un metodo statico `Raz uno()` che ritorna il razionale 1.

   Definire un esempio di `main()` che usi tutti i metodi della classe.

4. Il seguente programma compila.

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

class N {
public:
  string s;
  N* next;
  N(string t, N* p): s(t), next(p) { }
};
class C {
public:
  N* pointer;
  C(): pointer(new N("zorro",0)) {}
  void F(string t1, string t2 = "pippo") {
    pointer = new N(t1,pointer); pointer = new N(t2,pointer);
  }
  void G() {if(pointer) {N* p = pointer; pointer = pointer->next; delete p;}}
};
void Fun(C* p1, C* p2) {if(p1 != p2) {*p1 = *p2; p1->G();}}
void stampa(N* p) {if(p) {cout << p->s << ' '; stampa(p->next);} }

main(){
  C* p = new C; p->F("pluto");
  C* q = new C; q->F("qui","quo");
  Fun(p,q);
  stampa(p->pointer); cout << "*** "; stampa(q->pointer);
}
```

   Quale delle seguenti affermazioni è **VERA** per l'esecuzione del programma?

   (a) termina correttamente e stampa: `qui zorro *** quo qui zorro`

   (b) termina correttamente e stampa: `qui zorro *** qui zorro`

   (c) termina correttamente e stampa: `qui zorro ***`

   (d) provoca "segmentation fault"

5. Si consideri il seguente programma.

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

class It {
  friend class C;
public:
  bool operator<(It i) {return index < i.index;}
  It operator++(int) { It t = *this; index++; return t; }
  It operator+(int k) {index = index + k; return *this; }
private:
  int index;
};
class C {
public:
  C(int k) {if (k>0) {dim=k; p = new int[k];} for(int i=0; i<k; i++) *(p+i)=i;}
  It begin() { It t; t.index = 0; return t; }
  It end() { It t; t.index = dim; return t; }
  int& operator[](It i) {return *(p + i.index);}
private:
  int* p;
  int dim;
};

main() {
  C c1(4), c2(8);
  for(It i = c1.begin(); i < c1.end(); i++) cout << c1[i] << ' '; cout << "UNO\n";
  It i = c2.begin();
  for(int n=0; i < c2.end(); ++n, i = i+n) cout << c2[i] << ' '; cout << "DUE\n";
}
```

   (a) Non compila

   (b) Compila e l'esecuzione provoca le stampe:

   ..... `UNO`

   ..... `DUE`

6. Si considerino le seguenti dichiarazioni e definizioni:

```
class Nodo {
private:
  Nodo(string st="***", Nodo* s=0, Nodo* d=0): info(st), sx(s), dx(d) {}
  string info;
  Nodo* sx;
  Nodo* dx;
};
class Tree {
public:
  Tree(): radice(0) {}
  Tree(const Tree&); // dichiarazione costruttore di copia
private:
  Nodo* radice;
};
```

   Quindi, gli oggetti della classe `Tree` rappresentano *alberi binari ricorsivamente definiti di stringhe*. Si ridefinisca il costruttore di copia di `Tree` in modo che esegua copie profonde. Scrivere esplicitamente eventuali dichiarazioni `friend` che dovessero essere richieste da tale definizione.