Laurea in Informatica - Programmazione ad Oggetti - Appello d'Esame 25/01/2024

Esercizio Cosa Stampa

```
class A {
                                                                                                                      class B: virtual public A (
public:
                                                                                                                      public:
   A() {cout<< " A() ";}
~A() {cout<< " ~A ";}
                                                                                                                          B() {cout<< " B() ";}
                                                                                                                          virtual B() {cout<< " B ";}
                                                                                                                         virtual 'B() {cout<< " B ";}
virtual void g() const {cout << " B::g ";}
virtual const B* j() {cout << " B::j "; n(); return this;}
void k() {cout << " B::k "; j(); m(); }
void m() {cout << " B::m "; g(); j();}
virtual A& n() {cout << " B::n "; return *this;}</pre>
   A(const A& x) {cout<< " Ac ";}
   A(const A& x) {cout<< " A::j "; return this;}
virtual void k() {cout << " A::k "; m();}</pre>
    void m() {cout <<" A::m "; j();}</pre>
class C: virtual public B {
                                                                                                                      class D: virtual public B {
public:
                                                                                                                      public:
   C() {cout<< " C() ";}
-C() {cout<< " -C ";}
                                                                                                                          D() {cout<< " D() ";}
~D() {cout<< " ~D ";}
   void g() const {cout <<" C::g ";}</pre>
                                                                                                                          virtual void g() {cout <<" D::g ";}</pre>
   void k() override {cout <<" C::k "; B::n();}
virtual void m() {cout <<" C::m "; g(); j();}
B& n() override {cout <<" C::n "; return *this;}</pre>
                                                                                                                         const B* j() (cout <<" D::j "; return this;)
void k() const {cout <<" D::k "; k();}
void m() {cout <<" D::m "; g(); j();)</pre>
class E: public C, public D {
public:
   E() {cout<< " E() ";}
-E() {cout<< " -E ";}
   E(const E& x) {cout<< " Ec ";}
   virtual void g() const (cout <<" E::g ";)
const E* j() (cout <<" E::j "; return this;)
void m() (cout <<" E::m "; g(); j();)</pre>
   D& n() final {cout <<" E::n "; return *this;}
A* p1 = \text{new E}(); B* p2 = \text{new C}(); A* p3 = \text{new D}(); B* p4 = \text{new E}(); const A* p5 = \text{new D}(); const B* p6 = \text{new E}(); const E* p7 = \text{new E}();
```

Le precedenti definizioni compilano correttamente. Per ognuna delle seguenti istruzioni scrivere nell'apposito spazio:

- NON COMPILA se la compilazione dell'istruzione provoca un errore;
- UNDEFINED se lo statement compila correttamente ma la sua esecuzione provoca un undefined behaviour o un errore run-time;
- se l'istruzione compila correttamente e non provoca errori a run-time allora si scriva la stampa che l'esecuzione produce in output su cout; se non provoca alcuna stampa allora si scriva NESSUNA STAMPA.

(p1->j())->k();
(dynamic_cast <const e*="">(p1->j()))->g();</const>
p2->m();
(p2->j())->g();
p3->k();
(p4->n()).m();
((dynamic_cast <d*>(p4))->n()).k();</d*>
(dynamic_cast <e*>(p5))->j();</e*>
(dynamic_cast <e*>(const_cast<b*>(p6)))->k();</b*></e*>
new E(*p7);
delete pl;
delete p4;