Nume ..... Nr. 1 23 iunie 2025

Grupa ..... Programare orientată pe obiecte

Examen scris

Detalii: 1 punct din oficiu, 18 probleme fiecare valorează 0.5 puncte, timp de lucru 2 ore.

I. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneți o (singură) modificare prin care programul devine corect.

```
1.  #include<iostream>
2.  using namespace std;
3.  class Base {
4.  public:
5.  int f() const { cout << "B1\n"; return 1; }
6.  int f(string) const {
7.  cout << "B2\n"; return 1; } };
8.  class Derived : public Base {
9.  public: int f() const {
10. cout << "D1\n"; return 2; } };
11. int main() {
12. string s("hello");
13. Derived d;
14. int x = d.f();
15. d.f(s);
16. return 0; }
```

Programul compilează? DA ☐ NU ☐
Dacă DA ce se afișează pe ecran:

Dacă NU: de ce nu?

modificarea care îl face să meargă (o singură linie modificată, precizat nr linie modificată şi modificarea)

II. Descrieţi folosirea pointerilor folosiți împreună cu cuvântul cheie const (sintaxă, proprietăți, particularități, exemplu).

III. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneți o (singură) modificare prin care programul devine corect.

```
1.  #include <iostream>
2.  using namespace std;
3.  class Cls
4.  { int a;
5.  public:
6.  Cls(int x):a(x){cout<<a<<" ";}
7.  ~Cls(){cout<<a<<" ";}
8.  } A(10);
9.   void adauga() { static Cls B(20);}
10.       static Cls F(70);
11.       Cls C(30);
12.  int main(){Cls D(40);
13.  adauga();
14.  Cls E(50);
15.  static Cls G(80);
16.   cout<<"* "; return 0;}
```

Programul compilează? DA ☐ NU ☐
Dacă DA ce se afișează pe ecran:

Dacă NU: de ce nu?

modificarea care îl face să meargă (o singură linie modificată, precizat nr linie modificată şi modificarea)

IV. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneţi o (singură) modificare prin care programul devine corect.

```
1.  #include <iostream>
2.  using namespace std;
3.  class Baza{ public: Baza(){cout<<"CB\n";} };
4.  class Derivata1 : public Baza {
5.      public: Derivata1(){cout<<"CD1\n";}
6.              ~Derivata1(){cout<<"DD1\n";}    };
7.  class Derivata2 : public Baza {
8.      public:     Derivata2(){cout<<"CD2\n";}
9.          virtual ~Derivata2(){cout<<"DD2\n";} };
10. class Derivata3 : virtual public Baza{
11. public: Derivata3(){cout<<"CD3\n";}  };
12. class Derivata4 : public Baza {public:Derivata4(){cout<<"DD4\n";} };
13. class Derivata5 : public Derivata1, Derivata2, protected Derivata3,
14.    public Derivata4 { public: Derivata5(){cout<<"Derivata5\n";} };
15. int main(){Derivata5 ob; }
```

Programul compilează? DA ☐ NU ☐
Dacă DA ce se afişează pe ecran:

Dacă NU: de ce nu?

modificarea care îl face să meargă (o singură linie modificată, precizat nr linie modificată şi modificarea)

V. Descrieţi particularităţile unui constructor definit cu atributul protected sau private (sintaxă, proprietăţi, particularităţi, exemplu).

VI. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneţi o (singură) modificare prin care programul devine corect.

```
1.  #include <iostream>
2.  using namespace std;
3.  class I { public:I(){cout<<"CI\n";}
4.            I(int x){cout<<x<<"ci\n";}
5.            void afis(){cout<<"i\n";} };
6.  class uni : virtual public I{
7.  public: uni(){cout<<"CU\n";}
8.   uni(int x):I(x){cout<<x<<" cu\n";}
9.  void afis(){cout<<"u \n";} };
10. class oras : virtual public I{
11. public: oras(){cout<<"CO\n";}
12.         oras(int x):I(x){cout<<x<<" co\n";}
13. void afis(){cout<<"Bucuresti";} };
14. class unibuc : public uni,public oras{ public: unibuc(){cout<<"CUB\n";}
15. unibuc(int x):I(x){} };
16. int main(){
17. unibuc ob;
18. ob.afis();
19. unibuc ob2(10);}
```

Programul compilează? DA ☐ NU ☐
Dacă DA ce se afişează pe ecran:

Dacă NU: de ce nu?

modificarea care îl face să meargă (o singură linie modificată, precizat nr linie modificată şi modificarea)

VII. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneţi o (singură) modificare prin care programul devine corect.

```
1.  #include <iostream>
2.  using namespace std;
3.  class Cls{ public: void afis(){cout<<"1 ";}
4.  Cls operator+(Cls ob){cout<<"2 "; return Cls();}
5.  Cls operator-(Cls ob){cout<<"3 "; return Cls();}};
6.  class Cls2 : public Cls{
7.  public: void afis(){cout<<"4 ";}
8.  Cls2 operator+(Cls2 ob){cout<<"5 "; return Cls2();} };
9.  int main() {
10. Cls a;
11. Cls2 d,e,f;
12. (d + e).afis();
13. (d - e).afis();
14. (a + d).afis();
15. (d + a).afis();
16. }
```

Programul compilează? DA ☐ NU ☐
Dacă DA ce se afişează pe ecran:

Dacă NU: de ce nu?

modificarea care îl face să meargă (o singură linie modificată, precizat nr linie modificată şi modificarea)

VIII. Descrieţi particularităţile metodelor statice considerând în special folosirea lor la moştenirea multiplă. (sintaxă, proprietăţi, particularităţi, apelare).

IX. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneţi o (singură) modificare prin care programul devine corect.

```
1.  #include <iostream>
2.  using namespace std;
3.  int main(){
4.  const char* sFirst = "Examen\0 2025\n";
5.  char sSecond[64];
6.  const char* sSrc = sFirst;
7.  char* sDst = sSecond;
8.  while (*sDst++ = *sSrc++);
9.  std::cout << sSecond;
10. }
```

Programul compilează? DA ☐ NU ☐
Dacă DA ce se afişează pe ecran:

Dacă NU: de ce nu?

modificarea care îl face să meargă (o singură linie modificată, precizat nr linie modificată şi modificarea)

X. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneţi o (singură) modificare prin care programul devine corect.

```
1.   #include<iostream>
2.   using namespace std;
3.   int main()
4.   {int a = 1;
5.    int b = ++a;
6.    const int *a_ptr = &a;
7.    int *const b_ptr = &b;
8.    *b_ptr += 4;
9.    *a_ptr += 5;
10.   std::cout << *b_ptr << std::endl;
11.   std::cout << *a_ptr << std::endl;
12.  return 0; }
```

Programul compilează? DA ☐ NU ☐
Dacă DA ce se afişează pe ecran:

Dacă NU: de ce nu?

modificarea care îl face să meargă (o singură linie modificată, precizat nr linie modificată şi modificarea)

XI. Descrieţi particularităţile operatorului typeid. (sintaxă, proprietăţi, particularităţi, motivaţie).

XII. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneţi o (singură) modificare prin care programul devine corect.

```
1.  #include <iostream>
2.  using namespace std;
3.  class Baza{ public: virtual ~Baza(){};
4.  class D1 : virtual public Baza{};
5.  class D2 : public D1{};
6.  class D3 : virtual public Baza{};
7.  class D4 : public D1, public D3{};
8.  int main() {
9.  D4 ob;
10. Baza& re = ob;
11. try{ throw re; }
12. catch(D1& o){cout<<"D1\n";}
13. catch(D2& o){cout<<"D2\n";}
14. catch(D3& o){cout<<"D3\n";}
15. catch(D4& o){cout<<"D4\n";}
16. catch(Baza& o){cout<<"Baza\n";}
17. }
```

Programul compilează? DA ☐ NU ☐
Dacă DA ce se afişează pe ecran:

Dacă NU: de ce nu?

modificarea care îl face să meargă (o singură linie modificată, precizat nr linie modificată şi modificarea)

XIII. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneţi o (singură) modificare prin care programul devine corect.

```
1.  #include<iostream>
2.  using namespace std;
3.  class B { int b;
4.  public: B(int p = 1) { b = p; } };
5.  class D : public B { int* d;
6.  public:     D(int p) { d = new int; *d = p; }
7.  D(const D& s) : B(s) { d = new int; *d = *(s.d); }
8.      ~D() { delete d; }
9.      void set(int p) { *d = p; }};
10.
11. int main() {
12.     D o1(2), o2(3);
13.     o1 = o2;
14.     o2.set(4);
15.     return 0;}
```

Programul compilează? DA ☐ NU ☐
Dacă DA ce se afişează pe ecran:

Dacă NU: de ce nu?

modificarea care îl face să meargă (o singură linie modificată, precizat nr linie modificată şi modificarea)

XIV. Descrieţi noţiunea de destructor virtual pur în C++. (sintaxă, proprietăţi, particularităţi, motivaţie).

XV. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneţi o (singură) modificare prin care programul devine corect.

```
1.  #include <iostream>
2.  using namespace std;
3.  class B {public:
4.  virtual B* fv() { return this; }
5.  int adun(int p) { return p + 1; }};
6.  class D : public B {public:
7.  virtual D* fv() { return this; }
8.  int adun(int p) { return p + 2; }};
9.  int main() {
10. B* p = new D;
11. int x = p->fv()->adun(1);
12. std::cout << x << "\n";
13. return 0;}
```

Programul compilează? DA ☐ NU ☐
Dacă DA ce se afişează pe ecran:

Dacă NU: de ce nu?

modificarea care îl face să meargă (o singură linie modificată, precizat nr linie modificată şi modificarea)

XVI. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, *în caz negativ* propuneţi o (singură) modificare prin care programul devine corect.

```
1.  #include<iostream>
2.  using namespace std;
3.  class C {int c;
4.  public: C(int p = 1) { c = p; }
5.          int& get() const { return c; } };
6.
7.  int f(C op) { return op.get(); }
8.
9.  int main() {
10. C o1;
11. int x = f(o1);
12. std::cout << x << "\n";
13. return 0; }
```

Programul compilează? DA ☐ NU ☐
Dacă DA ce se afișează pe ecran:

Dacă NU: de ce nu?

modificarea care îl face să meargă (o singură linie modificată, precizat nr linie modificată şi modificarea)

XVII. Definirea copy-constructorului de către programator. (sintaxă, proprietăţi, particularităţi, motivaţie).

XVIII. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneţi o (singură) modificare prin care programul devine corect.

```
1. #include <iostream>
2. using namespace std;
3. class C { int a;
4.           static int x;
5. public:
6.       C(int a = 22) { x++; this->a=a; }
7.       static int f() {return x;}
8.       int getA(){return a;}
9. }a;
10.int C::x=20;
11.int main() {
12.   C a(33),b;
13.   cout<<"instantieri C: "<<C::f()<<" val elem:"<<::a.getA();
14.   return 0; }
```

Programul compilează? DA ☐ NU ☐
Dacă DA ce se afișează pe ecran:

Dacă NU: de ce nu?

modificarea care îl face să meargă (o singură linie modificată, precizat nr linie modificată şi modificarea)