

Examen scris

- I. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{
    int x;
    static int y;
public: A(int i, int j):x(i), y(j) { }
    int f() const; };
int A::y;
int A::f() const { return y; }
int main()
{ const A a(21,2);
  cout<<a.f();
  return 0;
}
```

REZ: Un camp static nu se poate initializa in lista de initializare a constructorului

- II. Descrieți pe scurt cum este implementată moștenirea virtuală și scopul în care este folosită.

- III. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A;
class B
{ int x;
  public: B(int i=107) { x=i; }
  operator A(); };
B::operator A() { return x; }
class A
{ int x;
  public: A(int i=6) { x=i; }
  int get_x() { return x; } };
int main()
{ B b;
  A a=b;
  cout<<a.get_x();
  return 0;
}
```

REZ: cls1 nu poate declara operator cls2() din cauza ordinii de declarare a claselor

- IV. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{
    int *v, dim;
public:
    A(int i) { dim=i; v=new int[dim];
              for (int j=0; j<dim; j++) v[j]=j; }
    A(A& a)  { dim=a.dim; v=new int[dim];
              for (int j=0; j<dim; j++) v[j]=a.v[j]; }
    ~A() { delete[] v; }
    int size() { return dim; }
    int& operator[](int i) { return v[i]; }
    A operator+(A a1); };
A A::operator+(A a1)
{
    A a2(0);
    a2.dim=dim; v=new int[a2.dim];
    for (int j=0; j<a2.dim; j++) a2.v[j]=v[j]+a1.v[j];
    return a2; }
ostream& operator<<(ostream& o, A a)
{
    for (int i=0; i<a.size(); i++) cout<<a[i]<<" ";
    return o; }
int main()
{
    A a(10), b(10), c(10);
    c=a+b;
    cout<<c;
    return 0;
}
```

REZ: nedeterminat

- V. Spuneți ce este obiectul implicit al unei metode și descrieți pe scurt proprietățile pe care le cunoașteți despre acesta.

- VI. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{
    int x;
public:
    A(int i=0):x(i) { }
    int get_x() { return x; }
    int& set_x(int i) { x=i; } };
A operator=(A a1, A a2)
{
    a1.set_x(a2.get_x());
    return a2;
}
int main()
{
    A a(212), b;
    cout<<(b=a).get_x();
    return 0;
}
```

REZ: Supraincercarea op. = in afara clasei

- VII. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{
    public: int x;
           A(int i=0) { x=i; }
           virtual A minus() { return(1-x); } };
class B: public A
{
    public: B(int i=0) { x=i; }
           void afisare() { cout<<x; } };
int main()
{
    A *p1=new B(18);
    *p1=p1->minus();
    p1->afisare();
    return 0;
}
REZ: p1 nu are metoda afisare()
```

- VIII. Descrieți pe scurt funcțiile virtuale și scopul în care sunt folosite.

- IX. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{
    int x, *y;
    public: A(int i) { x=i; y=new int[x]; }
           A(A& a) { x=a.x; y=new int[x]; }
           int get_x() const { return x; } };
int f(A a) { return a.get_x(); }
int main()
{
    const A a(5);
    cout<<(a.get_x()==f(a));
    return 0;
}
REZ: Un obiect const nu poate fi transferat prin ref. nonconst (in A(A&))
```

- X. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{
    protected: int x;
    public:     A(int i=14) { x=i; } };
class B: A
{
    public: B(B& b) { x=b.x; }
           void afisare() { cout<<x; } };
int main()
{
    B b1, b2(b1);
    b2.afisare();
    return 0;
}
```

REZ: clasa B nu are constructor B()

- XI. Descrieți pe scurt funcțiile statice și scopul în care sunt folosite.

- XII. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{ int *v; int x;
  public: A(int i=0):v(new int[i]),x(i) { for(int j=0; j<i; j++) v[j]=j; }
        int get_x() const { return x; }
        int& set_x(int i) { x=i;}
        A& operator=(A& a)
        { x=a.x;
          v=new int[x];
          for(int j=0; j<x; j++) v[j]=a[j];
          return a; }
        int& operator[](int i) const { return v[i]; } };
ostream& operator<<(ostream& o, const A& a)
{ for (int i=0; i<a.get_x(); i++) cout<<a[i]<<" ";
  return o; }
int main()
{ A a(12), b;
  cout<<(a=b=a);
  return 0;
}
```

REZ: nedeterminat

XIII. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează pentru o valoare întreagă citită egală cu -35, în caz negativ spuneți de ce nu este corect.

```
#include <iostream.h>
int f(float y)
{ if (y<0) throw y;
  return y/2;
}
int main()
{ int x;
  try
  {
    cout<<"Da-mi un numar par: ";
    cin>>x;
    if (x%2) x=f(x);
    else throw x;
    cout<<"Numarul "<<x<<" e bun!"<<endl;
  }
  catch (int i)
  { cout<<"Numarul "<<i<<" nu e bun!"<<endl;
  }
  return 0;
}
```

REZ: Abnormal program termination

XIV. Descrieți pe scurt diferența dintre transferul parametrilor unei funcții prin valoare și prin referință constantă.

XV. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include <iostream.h>
template<class T>
T f(T x, T y)
{ return x+y;
}
int f(int x, int y)
{ return x-y;
}
int main()
{ int *a=new int(3), b(23);
  cout<<f(a,b);
  return 0;
}
```

REZ: Nu exista operator+ pentru int,int\*

XVI. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{ int x;
  public: A(int i=0) { x=i; }
         A operator+(const A& a) { return x+a.x; }
         template <class T> ostream& operator<<(ostream&); };
template <class T>
ostream& A::operator<<(ostream& o) { o<<x; return o; }
int main()
{ A a1(33), a2(-21);
  cout<<a1+a2;
  return 0;
}
```

REZ: << nu este apelat cu sintaxa coresp. definitiei

XVII. Descrieți pe scurt cum se comportă destructorii la moștenire.

XVIII. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{ protected: int x;
  public: A(int i=-31) { x=i; }
         virtual A operator+(A a) { return x+a.x; } };
class B: public A
{ public: B(int i=12) { x=i; }
       B operator+(B b) { return x+b.x+1; }
       void afisare(){ cout<<x; } };
int main()
{ A *p1=new B, *p2=new A;
  B *p3=new A(p2->operator+(*p1));
  p3->afisare();
  return 0;
}
```

REZ: Nu se poate instantia p3 (B nu e baza pentru A)