Nume .....
Grupa .....

3 septembrie 2013
Programare orientate pe obiecte

Examen scris

I. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ spuneţi de ce nu este corect.

```
#include <iostream>
using namespace std;
class A
{   static int x;
    int y;
    public: A(int i) { x=i; y=-i; }
            static int put_x(A a) { return x+a.y; } };
int A::x=5;
int main()
{   A a(7);
    cout<<A::put_x(3);
    return 0; }
```

R: 0

II. Descrieţi pe scurt obiectul implicit al unei metode.

0,1 Obiectul care lanseaza o metoda in executie
0,1 Este transmis in metoda prin referinta (constanta daca metoda e constanta)
0,1 Poate fi accesat cu pointerul this
0,1 Metodele statice nu au obiect implicit
0,1 daca nu a scris prostii (dar a scris ceva!)

III. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afisează pentru o valoare citita egala cu -5, în caz negativ spuneţi de ce nu este corect.

```
#include <iostream>
#include <cmath>
using namespace std;
int f(int y, int z)
{ cout<<" y nu este referinta "; if (y<z) throw sqrt(z-y); return y/2;}
int f(int &y)
{ cout<<" y este referinta "; return y/2 ;}
int main()
{ int x;
  try
  { cout<<"Da-mi un numar par: ";
    cin>>x;
    if (x%2) x=f(x, 0);
    else x=f(x);
    cout<<"Numarul "<<x<<" e bun!"<<endl;  }
  catch (int i)
  { cout<<"Numarul "<<i<<" nu e bun!"<<endl;
  }  return 0; }
```

R: Se arunca o exceptie float care nu e prinsa de catch(int)

IV. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ spuneţi de ce nu este corect.

```
#include<iostream>
using namespace std;
class B
{     protected: int x;
      public: B(int i=16) { x=i; }
              virtual B f(B ob) { return x+ob.x; }
              void afisare(){ cout<<x; } };
class D: public B
{     public: D f(D ob) { return x+1; } };
int main()
{ B *p1=new D, *p2=new B, *p3=new B(p1->f(*p2));
        p3->afisare();
  return 0;
}
```

R: Nu exista conversie de la int la D la intoarcerea rezultatului in f()

V. Descrieţi pe scurt în ce constă polimorfismul de execuţie folosind metode virtuale.

0,1 sintaxa
0,1 metode virtuale
0,1 mostenire
0,1 instantiere dinamica
0,1 daca nu a scris prostii (dar a scris ceva!)

VI. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ spuneţi de ce nu este corect.

```
#include <iostream>
using namespace std;
class X {    int i;
public:    X(int ii = 5) { i = ii; cout<< i;};
           void afisare(int j) { cout<<i;}; }o;
int main()
{ X O (7);
  X &O2=o;
  O2.afisare(5);
  const X* p=&O;
  p->afisare(3);
  return 0; }
```

neconst const

R: p->afisare() e o functie const apelata de un obiect ~~neconstant~~

VII. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ spuneţi de ce nu este corect.

```
#include <iostream>
using namespace std;
class cls
{     int x;
public: cls(int i) { x=i; }
int set_x(int i) { int y=x; x=i; return y; }
int get_x(){ return x; } };
int main()
{ cls *p=new cls[10];
```

```
        int i=0;
        for(;i<10;i++) p[i].set_x(i);
        for(i=0;i<10;i++) cout<<p[i].get_x();
    return 0;
}
```

R: cls nu are constructor fara parametri

VIII. Descrieți pe scurt cum se realizează moștenirea unei clase și care este rolul ei în programarea orientată pe obiecte.

0,1 sintaxa

0,1 definitie (obiectul derivat se formeaza din elementele mostenite + elemente specifice)

0,1 acces

0,1 rolul

0,1 daca nu a scris prostii (dar a scris ceva!)

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>
using namespace std;
class cls1
{   int x;
    public: cls1(int i=0) { x=i; }
            cls1 operator+(const cls1& a) { return cls1(x+a.x); } };
      template <class T>
class cls2
{   T y;
    public: cls2() {}
            cls2(T i) { y=i; }
            cls2 operator+(cls2 a) { return cls2(y+a.y); }
void afisare(){ cout<<y; } };
int main()
{   cls2<int> a(5), b(18);
    cls2<cls1> c, d;
    (a+b).afisare();
    (c+d).afisare();
    return 0;
}
```

R: << un e def. pt. cls1

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>
using namespace std;
class A
{   int x;
    public: A(int i):x(i){}
            ~A(){ cout<<x; }
};
class B:public A
{   int y;
    public: B(int i,int j):A(i),y(j){}
            ~B(){ cout<<y; }
};
```

```
int main()
{   A *p=new B(7,-3);
    delete p;
    return 0;
}
```

R: 7

XI. Descrieţi pe scurt cum se implementează metodele de conversie între tipuri de date.

0,2 sintaxa + constructor de conversie
0,2 sintaxa + supraincarcarea op cast
0,1 daca nu a scris prostii (dar a scris ceva!)

XII. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ spuneţi de ce nu este corect.

```
#include <iostream>
using namespace std;
class cls
{ int x;
  public: cls(int i=13) {x=i; }
  int get_x() { return x; }
  int set_x(int i) { x=i;return x;}
  cls operator=(cls a1) { set_x(a1.get_x()); return a1;}
} ;
int main()
{ cls a=19,b;
  cout<<(a=b).get_x();
  return 0; }
```

R: 13

XIII. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ spuneţi de ce nu este corect.

```
#include <iostream>
#include <typeinfo>
using namespace std;
class B
{ int i;
  public: B() { i=1; }
          int get_i() { return i; } };
class D: public B
{ int j;
  public: D() { j=2; }
          int get_j() {return j; } };
int main()
{ B *p=new D;
  cout<<p->get_i();
  if (typeid(p)==typeid(D*)) cout<<((D*)p)->get_j();
  return 0; }
```

R: 12

XIV. Descrieţi pe scurt constructorul de copiere şi situaţiile în care este folosit.

0,1 sintaxa
0,1 varianta implicita

0,1 initializare
0,1 transmiterea parametrilor + intoarcerea rezultatului prin valoare
0,1 daca nu a scris prostii (dar a scris ceva!)

XV. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ spuneţi de ce nu este corect.

```
#include <iostream>
using namespace std;
struct X {   int i;
public:    X(int ii = 0) { i = ii; cout<< i;};
           void afisare() const {cout<<i;}; } ;
const X& apel(int j) {   return *(new X(j)); }

int main()
{ const X x(5);
  apel(2)=x;
  x.afisare();
  return 0; }
```

R: apel() intoarce referinta catre un obiect const care nu poate fi modificat printr-o atribuire

XVI. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ spuneţi de ce nu este corect.

```
#include <iostream>
using namespace std;
class A
{   static int *x;
    public: A() {}
            int get_x() { return (*x)++; } };
int *A::x(new int(1000));
int main()
{   A *a=new A,b;
    cout<<b.get_x();
    cout<<a->get_x();
    return 0; }
```

R: 10001001

XVII. Descrieţi pe scurt ce reprezintă şi cand se foloseşte transferul parametrilor prin referinţă constantă.

0,1 sintaxa
0,1 transmiterea prin referinta
0,1 diferenta cu const
0,1 de ce foloseste
0,1 daca nu a scris prostii (dar a scris ceva!)

XVIII. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ spuneţi de ce nu este corect.

```
#include<iostream>
using namespace std;
class A
```

```
{   int x;
    public: A(int i):x(i){}
            int get_x(){ return x; } };
class B: protected A
{   int y;
    public: B(int i,int j):y(i),A(j){}
            int get_y(){ return get_x()+y; } };
 class C: protected B
 {   int z;
     public: C(int i,int j,int k):z(i),B(j,k){}
             int get_z(){ return get_y()+z; } };
  int main()
  {   C c(1,2,3);
      cout<<c.get_z();
      return 0; }
```

R: 6