Nume ……..
Grupa …….

10 iunie 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(int i=x) { x=i;
; return x; } };
int A::x=5;
int main()
{   A a(7);
    cout<<A::put_x();
    return 0; }
```

R: Metoda statica put_x() incearca sa acceseze campul y nestatic

II. Spuneţi pe scurt ce este lista de iniţializare a unui constructor şi subliniaţi importanţa ei.

Sintaxa 0.1
Utilitate: initializarea campurilor si a constructorului bazei 0.1
Importanta pentru initializarea campurilor constante 0.2
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>
using namespace std;
int f(int y, int z)
{ cout<<" y nu este referinta "; if (y<z) throw 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: y nu e referinta Numarul 5 nu e bun!

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 X {int i;
public:
  X() { i = 0; }
  void set(int ii) { i = ii; cout<< ii+2;}
  int read() const { return i; }
  int permute() { return i = i * 47; } };
class Y : public X {
  int i;
  X j;
public:
  Y() { i = 0; j.set(2); }
  int change() {
          i = permute();
          j.permute();
          cout << j.read();
          return i; }
  void set(int ii) {
    i = ii;
    j.set(ii+1);
    X::set(ii); } };
int main() {
  Y D;
  D.change();
  D.read();
  D.permute();
  D.set(12);
  return 0; }
```

R: 4941514

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

Sintaxa 0.3 (Instantiere dinamica 0.1, Mostenire 0.1, Metode virtuale 0.1)
Descriere 0.1
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; }
```

R: afisare() nu e functie const

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 clsB {  int x;
  public: clsB(int i=0) {cout << "C0 "; x=i; }
          ~clsB(){ cout << "D0 ";} };
class cls :public clsB{ int x;
                clsB Y;
  public: cls(int i=0) {cout << "C1 "; x=i; }
          ~cls(){ cout << "D1 ";} } p;
class clss : virtual public clsB{ int x; cls xx;
  public: clss(int i=0) {cout << "C2 "; x=i; }
          ~clss(){ cout << "D2 ";} };
class clss2{   int x;
               clss xx;
               cls xxx;
  public: clss2(int i=0) {cout << "C3 "; x=i; }
          ~clss2(){ cout << "D3 ";} };
  clss o;
int main()
{ clss2 s;
  return 0;}
```

R: 0 0 1 0 0 0 1 2 0 0 0 1 2 0 0 1 3

VIII. Descrieţi pe scurt cele două feluri de folosire a cuvântului cheie “virtual” la moştenire şi în ce cazuri se folosesc.

Mostenire virtuala: Sintaxa 0.1, Utilitate 0.1
Functii virtuale: Sintaxa 0.1, Utilitate 0.1
0.1 daca nu a scris prostii (dar a scris ceva!)

IX. 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;
template<class T, class U>
T f(T x, U y)
{ return x+y; }
int f(int x, int y)
{ return x-y; }
int main()
{ int *a=new int(4), b(16);
  cout<<*f(a,b);
  return 0; }
```

R: afiseaza o valoare nedeterminata de la adresa lui a plus 16

X. 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;
template <class X> void swapt(X &a, X &b)
{   X temp;
    temp = a;
    a = b;
    b = temp; }
template <class X, class Y> void swapt(X &a, Y &b, Y c)
{   Y temp;
    temp = b;
    b = c;
    c = temp; }
void swapt(int &a, int b, int c=0)
{  int temp;
   temp = a+2;
   a = b-1;
   b = temp; }
int main()
{ int i=13,j=15;
  swapt(i,j);
  cout<< i<<j;
  swapt(i,j,12);
  cout<< i<<j;
  return 0; }
```

R: 14151415

XI. Cum se face supraîncărcarea operatorilor ca funcţii independente în C++. Particularităţi.

Sintaxa 0.1
Restrictii (minim 3 operatori care nu se pot supraincarca) 0.1
Acces la membrii privati ai clasei (prin friend) 0.1
Utilitate (supraincarcare << sau similar) 0.1
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 CL
{ static int x;
  public: CL(int i=3) {x=i; }
  int get_x() { return x; }
  int& set_x(int i) { x=i;return x;}
  CL operator=(CL a1) { set_x(a1.get_x()); return a1;}
} a(19);
int CL::x;
int main()
{ CL b;
  cout<<(b=a).get_x();
  return 0; }
```

R: 3

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

R: Baza B este inaccesibila pentru D (sau alternativ get_i este inaccesibil pentru noul obiect de tip D creat)

XIV. Ce reprezintă încapsularea şi cum se realizează?

Definitie (Date+Algoritmi) 0.1
Specificatori de acces 0.3
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(){cout<<i++;}; } o;
const X apel(int j) {   return X(j); }
void apel2(X& x) {
  x.afisare(); }
int main()
{ apel2(apel(2)=X(5));
  return 0; }
```

R: apel() intoarce un obiect const care e trimis prin referinta in apel2() (sau alternativ apare in membrul stang al unei atribuiri)

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
{   int *x;
    public: A():x(new int(1000)) {}
            int get_x() { x++; return *x; } };
int main()
{   A *a=new A,b;
    cout<<a->get_x()<<" "<<b.get_x();
    return 0; }
```

R:afiseaza nedeterminat de la adresa x+1

XVII. Să se descrie care e diferența dintre un pointer constant către un obiect și un pointer către un obiect constant.

Pointer constant (nu se poate modifica adresa) 0.2
Pointer catre obiect constant (nu se poate apela o metoda neconstanta) 0.2
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: public A
{   int y;
    public: B(int i,int j):y(i),A(j){}
            int get_y(){ return y; } };
class C: protected B
{   int z;
    public: C(int i,int j,int k):z(i),B(j,k){}
            int get_z(){ return z; } };
int main()
{   C c(1,2,3);
    cout<<c.get_x()+c.get_y()+c.get_z();
    return 0; }
```

R: c nu are acces la get_x() si nici la get_y()