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

20 iunie 2014
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.h>
class A
{ int i;
  protected: static int x;
  public: A(int j=7) {i=j; x=j;}
  int get_x() { return x; }
  int set_x(int j) { int y=x; x=j; return y; }
  A operator=(A a1) { set_x(a1.get_x()); return a1;}
};
int A::x=15;
int main()
{ A a(212), b;
  cout<<(b=a).get_x();
  return 0;
}
```

II. Cum se face supraîncărcarea operatorilor în C++ ca funcţii membre ale clasei. Particularităţi.

III. 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.h>
template<class X, class Y>
X f(X x, Y y)
{ return x+y;
}
int* f(int *x, int y)
{ return x-y;
}
int main()
{ int *a=new int(200), *b=a;
  cout<<*f(a,b);
  return 0;
}
```

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.h>
template <class X> void test(X &a, X &b)
{
    X temp;
    temp = a;
    a = b;
    b = temp;
    cout << "ttest\n"; }
void test(int &c, int &d)
{
   int temp;
   temp = c;
   c = d;
   d = temp;
   cout << "test 1\n"; }
void test(int e, int f)
{
   int temp;
   temp = e;
   e = f;
   f = temp;
   cout << "test 2\n"; }
void test(int g, int *h)
{  int temp;
   temp = g;
   g = *h;
   *h = temp;
   cout << "test 3\n"; }
int main()
{ int a=5, b=15, c=25, *d=&a;
  test(a,b);
  test(c,d);
  return 0; }
```

V. Descrieţi pe scurt intoarcerea rezultatului unei funcţii prin referinţă.

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>
class A
{     int valoare;
      public:     A(int param1=3):valoare(param1){}
                  int getValoare(){return this->valoare;}
};
int main()
{     A vector[] = {*(new A(3)), *(new A(4)), *(new A(5)), *(new A(6))};
      cout<<vector[2].getValoare();
      return 0;
}
```

VII. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează pentru o valoare întreagă citită egală cu 18, în caz negativ spuneţi de ce nu este corect.

```
#include <iostream>
int f(int y)
{ if (y<0) throw y;
  return y/2;
}
int main()
{ int x;
  try
  {
  cout<<"Da-mi un numar par: ";
  cin>>x;
  if (x%3) x=f(x);
  else throw x;
  cout<<"Numarul "<<x<<" e bun!"<<endl;
  }
  catch (int i)
  {  cout<<"Numarul "<<i<<" nu e bun!"<<endl;
  }
  return 0;
}
```

VIII. Descrieţi pe scurt moştenirea multiplă in C++.

pentru validare bonus 6

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.h>
class A
{ int i;
  public: A() { i=1; }
  virtual int get_i() { return i; }};
class B: public A
{ int j;
  public: B() { j=2; }
  int get_i() {return A::get_i()+j; }};
int main()
{ const int i = cin.get();
  if (i%3) { A o;}
  else {B o;}
  cout<<o.get_i();
  return 0;
}
```

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>
class cls
{ int x;
  public: cls(int i=0) {cout << " c1 "; x=i; }
          ~cls(){ cout << " d 1 ";}  };

class cls1
{  int x; cls xx;
  public: cls1(int i=0) {cout << " c2 "; x=i; }
          ~cls1(){ cout << " d2 ";} }c;

 class cls2
{ int x; cls1 xx; cls xxx;
  public: cls2(int i=0) {cout << " c3 "; x=i; }
           ~cls2(){ cout << " d3 ";}  };
int main()
{ cls2 s;
  return 0;
}
```

XI. Descrieti pe scurt definirea, implementarea şi folosirea funcţiilor virtuale

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>
#include <typeinfo>
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((B*)p).name()=="D*") cout<<((D*)p)->get_j();
  return 0;
}
```

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

```
#include <iostream>
using namespace std;
class cls
{     int x;
public: cls(int i=3) { x=i; }
        int& f() const { return x; } };
int main()
{ const cls a(-3);
        int b=a.f();
  cout<<b;
  return 0;
}
```

XIV. Descrieţi pe scurt ce reprezintă încapsularea şi moştenirea şi cum funcţionează împreună?

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>
class B
{     protected: int x;
      public:       B(int i=0) { x=i; }
              virtual B minus() { return(1-x); } };
class D: public B
{     public: D(int i=0):B(i) {}
              void afisare() { cout<<x; } };
int main()
{     D *p1=new D(18);
      *p1=p1->minus();
      p1->afisare();
    return 0;
}
```

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 B
{     protected: int x;
      public:    B(int i=14) { x=i; } };
class D: B
{     public: D(B ob) { x=ob.x; }
              void afisare() { cout<<x; } };
int main()
{     B ob1;
      D ob2(ob1);
      ob2.afisare();
      return 0;
}
```

XVII. Descrieţi pe scurt definirea şi utilizarea claselor generice (şablon).

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 B
{     protected: int x;
      public: B(int i=105) { x=i; }
              virtual get_x()=0;
class D: public B
{     public: D(int i):B(i) {}
              D operator+(const D& a) {return x+a.x; } };
int main()
{ D ob1(17), ob2(-12);
  cout<<(ob1+ob2).get_x();
  return 0;
}
```