Anul I Informatică (valabil şi pentru Anul II Matematică-Informatică) iunie 2005
Proiectare şi programare orientate pe obiecte

Examen scris

I. Spuneți de câte ori se execută fiecare constructor în programul de mai jos şi în ce ordine.

```
#include <iostream.h>
class cls1
{  protected: int x;
   public: cls1(){ x=13; }  };
class cls2: public cls1
{  int y;
   public: cls2(){ y=15; }
   int f(cls2 ob) { return (ob.x+ob.y); }  };
int main()
{  cls2 ob; cout<<ob.f(ob);
   return 0;
}
```

II. Descrieți pe scurt diferența dintre funcțiile care returnează valoare şi cele care returnează referință.

III. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afişează, altfel, spuneți de ce nu este corect.

```
#include <iostream.h>
class cls1
      {  int x;
         public: cls1(){ x=13; }
                 int g(){ static int i; i++; return (i+x); } };
class cls2
      {  int x;
         public: cls2(){ x=27; }
                 cls1& f(){ static cls1 ob; return ob; } };
int main()
{     cls2 ob;
      cout<<ob.f().g();
      return 0;
}
```

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

```
#include <iostream.h>
class cls1
{     protected: int x;
      public:    cls1(int i=10) { x=i; }
                 int get_x() { return x;} };
class cls2: cls1
{     public: cls2(int i):cls1(i) {} };
int main()
{     cls d(37);
      cout<<d.get_x();
      return 0;
}
```

V. Descrieți pe scurt cum se pot defini funcții de conversie între tipuri (clase).

VI. Cum trebuie definită variabila `n` din clasa de mai jos, dacă dorim ca ea să contorizeze numărul tuturor obiectelor care aparțin clasei `cls` la un moment dat.

```
class cls
{     int n;
      public: cls() { n ++; }
              ~cls(){ n --; }  };
```

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

```
#include <iostream.h>
class cls
{     int x;
      public: cls(int y) {x=y; }
              int operator*(cls a,cls b){return (a.x*b.x); } };
int main()
{     cls m(100),n(15);
      cout<<m*n;
      return 0;
}
```

VIII. Spuneți care este diferența dintre clasa generică (template) şi clasa abstractă şi]n ce situații se folosește fiecare dintre ele.

IX. Fiind date următoarele tipuri de clase:

```
class B { /* instructiuni */ };
class D1: virtual B { /* instructiuni */ };
class D2: virtual B { /* instructiuni */ };
class D3: B { /* instructiuni */ };
class D4: private B { /* instructiuni */ };
class D5: virtual public B { /* instructiuni */ };
class M1: D1, public D2, D3, private D4, virtual D5
{ /* instructiuni */ };
class M2: D1, D2, virtual D3, virtual D4, virtual D5
{ /* instructiuni */ };
```

spuneți de câte ori este moştenită clasa B în clasa M1. Dar în clasa M2 ? Justificați.

X. Spuneți care dintre declarațiile funcției `main()` sunt corecte în programul de mai jos. Justificați.

```
#include <iostream.h>
class B1 { public: int x; };
class B2 { int y; };
class B3 { public: int z; };
class B4 { public: int t; };
class D: private B1, protected B2, public B3, B4
{     int u; };
int main()
{     D d;
      cout<<d.u;
      cout<<d.x;
      cout<<d.y;
      cout<<d.z;
      cout<<d.t;
      return 0;
}
```

XI. Descrieți pe scurt diferența dintre transferul prin valoare şi transferul prin referință al parametrilor în cazul apelului unei funcții.

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

```
#include <iostream.h>
class cls
{             int vi;
      public: cls(int v=37) { vi=v; }
      friend int& f(cls); };
int& f(cls c) { return c.vi; }
int main()
{     const cls d(15);
      f(d)=8;
      cout<<f(d);
      return 0;
}
```

XIII. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, altfel, spuneți de ce nu este corect.

```
#include <iostream.h>
class cls1
{ public: int x;
          cls1(int i=20) { x=i; } };
class cls2
{ public: int y;
          cls2(int i=30) { y=i; }
          operator cls1() { cls1 ob; ob.x=y; return ob; } };
cls1 f(cls1 ob)
{ ob.x++;
  return ob;
}
int main()
{ cls1 ob1; f(ob1); cout<<ob1.x;
  cls2 ob2; f(ob2); cout<<ob2.y;
  return 0;
}
```

XIV. Spuneți pe scurt prin ce se caracterizează o metodă statică a unei clase.

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

```
#include <iostream.h>
class cls
{     int x;
      public: cls(int i=25) { x=i; }
              int f(); };
int cls::f() { return x; }
int main()
{     const cls d(15);
      cout<<d.f();
      return 0;
}
```

XVI. Spuneți de câte ori se apelează destructorul clasei `cls` în programul de mai jos. Justificați.

```
#include<iostream.h>
class cls
{  public: int x;
           cls(int i=0) { x=i; } };
cls f(cls c)
{     c.x++;
      return c;
}
int main()
{     cls r(10);
      cls s=f(r);
      return 0;
}
```

XVII. Spuneți care este diferența dintre incluziunea de clase şi moştenirea de clase şi când se foloseşte fiecare metodă.

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

```
#include <iostream.h>
template <class tip>
tip dif(tip x, tip y)
{ return x-y;
}
unsigned dif(unsigned x, unsigned y)
{ return x>=y?x-y:y-x;
}
int main()
{ unsigned i=7,j=8
  cout<<dif(i,j);
  return 0;
}
```