Anul II Matematică-Informatică ianuarie 2007
Proiectare şi programare orientate pe obiecte

Test scris (verificare)

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

```
class cls1
      {  protected: int x;
         public: cls1(){ x=13; }  };
class cls2: public cls1
      {  protected: int y;
         public: cls2(){ y=15; }  };
class cls3: public cls2
      {  protected: int z;
         public: cls3(){ z=17; }
         int f(cls3 ob){ return ob.x+ob.y+ob.z; }  };
int main()
{     cls3 ob;
      ob.f(ob);
      return 0;
}
```

II. Spuneți ce reprezintă o funcție virtuală şi în ce condiții o funcție virtuală defineşte o clasă abstractă.

III. 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 x;
      public: cls() { x=23; }
      int get_x(){ return x; } };
int main()
{     cls *p1, *p2;
      p1=new cls;
      p2=(cls*)malloc(sizeof(cls));
      int x=p1->get_x()+p2->get_x();
      cout<<x;
      return 0;
}
```

IV. 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 B
{     int a;
      B(int i=0) { a=i; }
      int get_a(){ return a; } };
class D: protected B
{     public: D(int x=0): B(x) {}
              int get_a() { return B::get_a(); } };
int main()
{     D d(-89);
      cout<<d.get_a();
      return 0;
}
```

V. Spuneți pe scurt prin ce se caracterizează un câmp static al unei clase.

VI. 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 cls1
{     public: int a;
              cls1() { a=7; } };
class cls2
{     public: int b;
              cls2(int i) { b=i; }
              cls2(cls1& x) { b=x.a; } };
int main()
{     cls1 x;
      cout<<x.a;
      cls2 y(x);
      cout<<y.b;
      return 0;
}
```

VII. 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
{     public: int sa;
              cls(int s=0) { sa=s; }
              operator float() { return sa; } };
int f(float c) { return (c*(1+c/100)); }
int main()
{     cls p(35);
      cout<<f(p);
      return 0;
}
```

VIII. Spuneți ce reprezintă o funcție prietenă (friend) a unei clase.

IX. 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 B
{ protected: int x;
  public: B() { x=78; } };
class D1: virtual public B
{ public: D1() { x=15; } };
class D2: virtual public B
{ public: D2() { x=37; } };
class C: public D2, public D1
{ public: int get_x() { return x; } };
int main()
{ C ob;
  cout<<ob.get_x();
  return 0;
}
```

X. 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>
class cls
{ tip z;
  public: cls(tip i) { z=i; }
          tip operator-(cls a) { return z-a.z; }
          int operator>=(cls a) { return z>=a.z; } };
template <class tip>
tip dif(tip x, tip y)
{ return x-y;
}
int dif(cls<int> x, cls<float> y)
{ return x>=y?x-y:y-x;
}
int main()
{ cls<int> i=3; cls<float> j=4;
  cout<<dif(i,j);
  return 0;
}
```

XI. Spuneți dacă o variabilă constantă poate fi transmisă ca parametru al unei funcții şi dacă da, în ce situații. Justificați.

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

```
class cls
{     int x;
public: cls(int i=0) { x=i; }
        cls(cls& ob) { x=ob.x; } };
cls& f(cls &c)
{     return c;
}
main()
{     cls r;
      cls s=r;
      f(f(f(s)));
}
```

XIII. 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 B
{             int x;
      public: B(int i=0) { x=i; } };
class D: public B
{     public: D():B(15) {}
              int f() { return x; } };
int main()
{     D d;
      cout<<d.f();
      return 0;
}
```

XIV. Descrieți pe scurt ce reprezintă obiectul implicit al unei metode.

XV. 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
{     static int i;
      int j;
      public: cls(int x=7) { j=x; }
              static int imp(int k){ cls a; return i+k+a.j; } };
int cls::i;
int main()
{ int k=5;
  cout<<cls::imp(k);
  return 0;
}
```

XVI. 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 x;
      public: cls(int i=32) { x=i; }
              int f() const { return x++; } };
void main()
{     const cls d(-15);
      cout<<d.f();
}
```

XVII. Descrieți pe scurt ce reguli verifică supraîncărcarea operatorilor.

XVIII. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afişează pentru o valoare întreagă citită egală cu 7, în caz negativ spuneți de ce nu este corect.

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