Universitatea "Al.I.Cuza" Iasi
Facultatea de Informatică

Nume:

An: I II III **Semian:** A B **Grupa:** 1 2 3 4 5 6 7

POO – Test 2 B-Barem
14.06.2010

Observaţii:
1. *Nu este permisă consultarea bibliografiei.* ***2.*** *Toate întrebările sunt obligatorii.* ***3.*** *Dacă nu este precizat altfel, fiecare întrebare este notată cu 3 puncte. Repartiţia punctelor la întrebările grilă este: 1 punct alegerea corectă a variantei, 2 puncte justificarea. Alegerea corectă se punctează numai dacă justificarea este total sau parţial corectă.* ***4.*** *Nu este permisă utilizarea de foi suplimentare.*

1) În codul de mai jos sunt folosite concepte C++: încapsulare, derivare, polimorfism, spații de nume, operator de alocare, constructori, destructori. Explicați 3 din aceste concepte și precizați locul unde au fost folosite. Care este rezultatul execuției programului?

```
#include <iostream>
using namespace std;
class A{
public:
      A(){cout << "A ";}
      virtual ~A(){cout << "~A ";}
};
class D{
public:
      D(){cout << "D ";}
      ~D(){cout << "~D ";}
};

class C:virtual public A, public D{
public:
      C(){cout << "C ";}
      ~C(){cout << "~C ";}
};

void main(){
      A* pA; pA = new C; delete pA;
};
```

Răspuns.

3x0.5p = 1.5p

main()(explicatii+rezultat) 1.5p

(Rezultatul : A D C ~C ~D ~A)

2) Ce va afișa programul de mai jos după execuție?

```
#include <iostream>
using namespace std;
struct punct{int x, y;};
punct operator~(const punct& p){
      punct q;
      q.x = -p.y;
      q.y = -p.x;
      return q;
}
ostream& operator<<(ostream& o, punct p){
      o << "(" << p.x << ", " << p.y << ")";
      return o;
}
int main(){
      punct alpha = { 11, 10 };
      cout << ~alpha  << endl;
      return 0;
}
```

a) `(00, 01)`
b) `(11, 10)`
c) `(-10, -11)` **CORECT**
d) nimic deoarece conține erori de sintaxă

Justificare.

Membrii struct sunt public: 0.5p
Supraincarcare operator~: 0.5p
Supraincarcare operator<< 0.5p
Concluzie 0.5p

3) Ce va afișa programul de mai jos după execuție?

```
#include <iostream>
using namespace std;
class A {
public:
      A(int j=0){i=j;cout<<" A ";}
      ~A(){cout<<" ~A ";}
```

a) `A B1 3 ~B ~A`
b) `A B2 3 ~B ~A` **CORECT**
c) `B2 A 3 ~A ~B`
d) `B2 A 0 ~A ~B`
e) nimic deoarece conține erori de sintaxă

```
      int get_i(){return i;}
protected:
      int i;
};
class B : public A {
public:
      B(){cout <<" B1 ";}
      B(int j):A(j){cout<<" B2 ";}
      ~B(){cout<<" ~B ";}
};
void main() {
      B ob = 3; A* p = &ob;
      cout << p->get_i();
}
```

Justificare.

A-baza, B- derivata 0.5p
Ordine de apel constructori in declaratia din main() 0.5p
p->get_i() 0.5p
Ordine de apel destructori 0.5p

4)

```
#include <iostream>
using namespace std;
class A{public:virtual char* f() = 0;};
class B:public A{
public:
  char* f(){cout << b; return "baza";}
protected:
  static int b;
};
class C:public B{
public:
  C(){b++;}
  char* f(){ cout << b; return "derivata";}
};
int B::b = 0;
int main() {
  A* a = new C;
  cout << a->f() << " ";
  B* b = new C;
  cout << b->f() << " ";
  delete a;delete b;return 0;
}
```

Ce va afişa programul alăturat?

a) `0derivata 1derivata`
b) `1derivata 2derivata` **CORECT**
c) `1baza 2baza`
d) `1baza 2derivata`

Justificare.

Ierarhia de clase 0.5p
f polimorfa 0.5p
b static 0.5p
obiectele a si b 0.5p

5) Explicaţi codul de mai jos şi precizați rezultatul execuţie.

```
#include <iostream>
using namespace std;
template <class T1, class T2> class myclass
{
public:
  myclass(T1 a, T2 b) {i = a; j = b;}
  void show(){
        cout << i << ' ' << j << '\n';
  }
private:
      T1 i; T2 j;
};
int main(){
  myclass<int, double> object1(10, 0.23);
  myclass<char, char*> object2('A',"test");
  object1.show();
  object2.show();
  return 0;
}
```

Răspuns.

Clasa myclass parametrizată 0.5p
Constructorul clasei 0.5p
object1 0.5p
object2 0.5p
Rezultatul 1p

6) Explicaţi codul de mai jos şi precizaţi rezultatul execuției

```
#include <iostream>
using namespace std;
void f(int n)
{
  try {
        if (n >= 0)throw "nenegativ; ";
        else throw "pozitiv; ";
  }
  catch(const char * s) {
        cout << "in f(): "  << s << ' ';
        throw ;
  }
}
int main()
{
  cout << "Start\n";
  try{
        f(11);
  }
  catch(const char * s) {
        cout << "in main: " << s << ' ' ;
  }
  cout << "End";
  return 0;
}
```

Răspuns.

Bloc try in f	0.5p
catch in f	0.5p
rethrow in f	0.5p
Bloc try in main	0.5p
catch în main	0.5p
rezultat	0.5p

```
Start
in f(): nenegativ;  in main: nenegativ;
End
```

7) Explicaţi codul de mai jos şi precizaţi rezultatul execuției

```
#include <iostream>
#include <queue>
#include <list>
using namespace std;
int main()
{
  int data[] = {21, 43, 23, 12, 15, 42};
  queue<int> s;
  for (int i = 1; i < 5; ++i)
        s.push(data[i]);
  s.pop(); cout << s.front()<< ' ';
  s.push(data[0]);
  cout << s.size() << endl;
  while (!s.empty()) {
    cout << s.front() <<' ';
    s.pop();
  }
  cout << s.size() << endl;
  return 0;
}
```

Răspuns.

23 4
23 12 15 21 0

S – coada de int implementată cu deque	0.5p
Conținut după for: 43 23 12 15	0.5p
top după pop: 23	0.5p
s.push(data[0]) 21	0.5p
dupa while: 23 12 15 21	0.5p
s.size() = 0	0.5p

8) (6 puncte)

Să se scrie codul C++ pentru implementarea unei clase `Distanta` a cărei instanțe sa fie reprezentări de distanțe măsurate în kilometri și metri. Un obiect al clasei poate fi: „25Km, 450m" dar nu „24Km,1450m" (numărul de metri este cel mult 999). Responsabilitățile clasei sunt:

	Declarația de clasa	0.5p
a)	inițializează o distanță de la două numere întregi pozitive	0.5p
b)	inițializează o distanță având ca intrare altă distanță(copiere);	1p
c)	atribuie o distanță altei distanțe (`operator=`);	1p
d)	afișează o distanță (`operator<<`)	1p
e)	aduna două distanțe (`operator+`).	2p