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

Nume:

An: I | II | III **Semian:** A | B **Grupa:** 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8

POO
26.06.12

Observaţii:
*1. Nu este permisă consultarea bibliografiei. **2.** Toate întrebările sunt obligatorii. **3.** 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) Explicați asemănările și deosebirile dintre relația de asociere și relația de agregare.
Răspuns.

2)

```
#include <iostream>
using namespace std;
class A {
public:
      A() { cout << "A() "; }
};
class B {
public:
      B() { cout << "B() "; }
};
class C:public A, public B {
public:
      C() { cout << "C() "; }
};
class E {
public:
      E() { cout << "E() "; }
      E(A una, B unb, C unc)
            : b(unb), a(una), c(unc)
      { cout << "E() "; }
private:
      C c; B b; A a;
};
int main() {
      E e;
      return 0;
}
```

Ce va afişa programul alăturat atunci când va fi executat?
a) `B() A() A() B() C() E()`
b) `A() B() C() B() A() E()`
c) `A() B() C() E() B() A()`
d) `E() C() A() B() B() A()`
e) nimic deoarece conţine erori de sintaxă
Justificare

3)

```
#include <iostream>
using namespace std;
class Patrat;
class Drepunghi {
public:
    int aria ()
      {return (latime * inaltime);}
```

Explicaţi programul alăturat şi precizaţi ce va afişa după execuţie?
Raspuns.

```
    void convert (Patrat a);
private:
      int latime, inaltime;
};
class Patrat {
public:
    void set_latura (int a)
      {latura=a;}
    friend class Drepunghi;
private:
      int latura;
};
void Drepunghi::convert (Patrat a) {
  latime = a.latura;
  inaltime = a.latura;
}
  int main () {
  Patrat p;
  Drepunghi d;
  p.set_latura(4);
  d.convert(p);
  cout << d.aria();
  return 0;
}
```

4)

```
#include <iostream>
using namespace std;
void f(int& x){
  if (x % 2)
    throw "impar";
  x = x / 2;
}
void g(int& x){
  if (!x)
    throw x;
  x = 15 / x;
}
int main(){
  int m = 26;
  try{
    f(m); g(m); f(m);
    cout << "m = " << m << endl;
  }
  catch (char *excChar)  {
        cout << "Exceptie: " << excChar;
  }
  catch (char excInt){
    cout << "Exceptie " << excInt;
  }
  return 0;
}
```

Ce va afişa programul alăturat atunci când va fi executat?

a) Exceptie 13
b) Exceptie 26
c) Exceptie impar
d) m = 0
e) m = 1

Justificare.

5)

```
#include <iostream>
#include <vector>
using namespace std;
template <class T>
class myelement {
public:
    myelement (T arg=0) {element=arg;}
    T increase () {return ++element;}
private:
    T element;
};
```

Explicaţi programul alăturat şi precizaţi ce va afişa după execuţie?

Răspuns.

```
template <>
class myelement <char> {
public:
    myelement (char arg='a') {element=arg;}
    char increase ()
    {
      if ((element>='a')&&(element<='z'))
      element+='A'-'a';
      return element;
    }
private:
    char element;
};
int main () {
  vector<myelement<int>> a(3);
  vector<myelement<char>> c(4, 'i');
  int x[]={1,3,5,7,9};
  char y[] = "info";
  a.assign(x+1,x+4);
  c.assign(y, y+4);
  cout << a[2].increase() << endl;
  cout << c[1].increase() << endl;
  return 0;
}
```

6) (12 puncte) Se dorește să se scrie o clasă Manager care să printeze informații despre vehiculele închiriate:

- dacă vehiculul este un automobil, atunci interesează numărul de locuri, numărul de kilometri parcursi, costul, stare (închiriat sau nu)
- dacă vehiculul este un camion, atunci interesează capacitatea, numărul de kilometri parcursi, costul, stare
- dacă vehiculul este o bicicletă, atunci interesează numărul de zile cât a fost închiriată, costul, stare

1. Să se deseneze diagrama claselor implicate (vor apărea cel puțin Manager, Automobil, Camion, Bicicleta)
2. Să se descrie pe scurt conceptele POO care sunt aplicate.
3. Să se scrie codul C++ minimal pentru implementarea diagramei.
4. Să se scrie o piesă de cod care creează o instanță a clase Manager compusă din 6 instanțe Automobil, Camion sau Bicicleta.