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 – Test scris A - Barem
04.06.2014

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.* ***4.*** *Nu este permisă utilizarea de foi suplimentare.*

1) Să se explice pe scurt ce înseamnă supraîncărcarea metodelor, suprascrierea metodelor și relația dintre ele.
Răspuns.
1p - supraîncărcare (metode cu același nume în aceeași clasă dar cu signaturi diferite)
1p - suprascriere (metode cu același nume și aceleași signaturi în clase diferite în cadrul unei ierarhii de clase)
1p - relația (amândouă sunt instanțe de polimorfism: prima polimorfism ad-hoc, cea de-a doua polimorfism universal (incluziune))

2)

```
#include <iostream>
using namespace std;
class Base1{
public:
      int m;
      Base1(int n=1): m(n)
      {cout << "B1:" << ++m;}
};
class Base2{
public:
      int m;
      Base2(int n=2): m(n)
      {cout << "B2:" << m;}
};
class Derived1: public Base1{
public:
      double d;
      Derived1(double de = 0.0): d(de)
      {cout << "D1:" << d;}
};
class Derived2: public Base2, public Derived1{
public:
      double d;
      Derived2(double de = 0.0): d(de)
      {cout << "D2:" << d;}
};
int main(){
      Derived2 c(2.5);
      return 0;
}
```

Construiţi diagrama ierarhiei de clase din codul alăturat, descrieți și explicați ordinea de apel al constructorilor și al destructorilor si precizați ce se afișează după execuție și cum se obține mesajul afișat (cine afișează fiecare componentă).
Răspuns.

- Diagrama 1p
- Ordinea de apel a cons/dest 1p
- Rezultatul corect (B2:2B1:2D1:0D2:2.5) 1p
  - 0.25 fiecare componenta din mesaj

3)

```
#include <iostream>
using namespace std;
template<typename T>class clsTemplate{
public:
      T value;
      clsTemplate(T i){this->value = i;}
      void test(){cout << value ;}
};
class clsChild : public clsTemplate<char>{
public:
      clsChild(): clsTemplate<char>( 65 ){}
      clsChild(char c): clsTemplate<char>(c){}
};
int main(){
```

În contextul programării generice, precizați care sunt definițiile parametrizate din codul alăturat și cum suntele utilizate. Care este rezultatul execuției programului și cum este obținut?
Răspuns.

- clasa parametrizată clsTemplate cu parametrul T (clasa) 1p
- utilizarea clasei parametrizate pentru definirea clasei clsChild 0.5p
- instanțierea claselor 0.5p
- rezultatul corect(42 A A) 1p

```
      clsTemplate <int> a(42);
      clsChild b('A'), c;
      a.test();b.test();c.test();
      return 0;
}
```

4)

```
#include <iostream>
using namespace std;
class A{public:virtual char* f() = 0;};
class B:public A{
public:
  char* f(){cout << b; return "B::f";}
protected:
  static int b;
};
class C:public B{
public:
  C(){b++;}
  char* f(){ cout << b; return "C::f";}
};
int B::b = 4;
int main() {
  A* a = new C;
  cout << a->f() << " ";
  B* b = new C;
  cout << b->f() << " ";
  delete a;delete b;return 0;
}
```

Explicați cum este implementat conceptul de polimorfism în codul alăturat. Care este rezultatul execuţiei programului și cum este obținut?

Răspuns.

5C::f 6C::f

- polimorfism
  - ierahia de clase 0.5p
  - suprascriere 0.5p
  - mecanism funcții virtuale 0.5p
- rezultat
  - `a->f()` 0.5p
  - `b->f()` 0.5p
  - b static 0.5p

5)

```
#include <iostream>
#include <stack>
#include <list>
using namespace std;
int main()
{
  int data[] = {14,31,12,21,15,51};
  stack<int, list<int>> s;
  for (int i = 1; i < 5; ++i)
        s.push(data[i]);
  s.pop(); cout << s.top()<< ' ';
  s.push(data[0]);
  while (!s.empty()) {
    cout << s.top() <<' ';
    s.pop();
  }
  cout << s.size() << endl;
  return 0;
}
```

Precizați ce containere din STL sunt utilizate, împreună cu o descriere scurtă a acestora, și explicați modul de utilizare a acestora în codul alăturat.

Răspuns.

21 14 21 12 31 0

- stiva s 0.5p
- populare stiva s 0.5p
- pop, push 0.5p
- while ... 1p
- afisare rezultat 0.5p

6) (12 puncte)

Clasa *Temperature* e responsabilă cu înregistrarea temperaturilor (numere întregi) din oră în oră pe ultimele șapte zile. Clasa *MinMax* e responsabilă cu afișarea temperaturii minime respectiv maxime pe ultimele șapte zile. Clasa *LastFour* e responsabilă cu afișarea ultimelor patru valori înregistrate.

a) Să se deseneze diagrama pentru șablonul *Observer* cu descrierea pe scurt a rolului fiecarui participant.

b) Să se scrie codul C++ care implementează metoda de notificare.

c) Să se proiecteze clasele *Temperature*, *MinMax*, *LastFour* utilizând șablonul Observer (se poate instanția doar diagrama sau descrie rolul celor trei clase și ce metode includ). Pentru metode se menționează numai signatura.

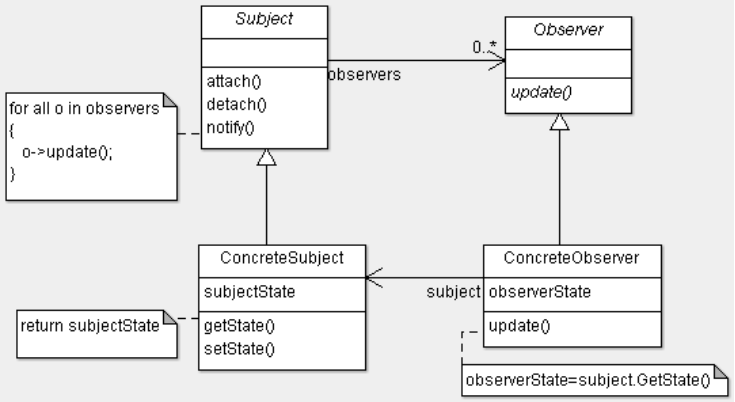
d) Să se scrie codul C++ pentru metodele de actualizare ale claselor *MinMax* și *LastFour*.

e) Să de scrie codul C++ care creează o instanță *Temperature* ce înregistrează temperaturile 1,2, ..., 36 și câte o instanță a claselor *MinMax* și *LastFour* care sunt în relație cu instanța *Temperature* și afișează valorile corespunzătoare.

Observație. Obiectele agregate vor fi implementate cu containere STL.

Răspuns .

a) 2p:



b) 2p

```
void Observer::notify() {
    list<Observer*>::iterator it;
    for (it=observers.begin(); it != observers.end(); ++it)
      (*it)->update();
  }
```

c) 2p similar lui a): Subject = Temerature, ConcreteObserver = MinMax, LastFour

d) 4p = 2p + 2p

```
void MinMax::update() {
    int temp = subject->getLast();
    if (min > temp) min = temp;
    if (max < temp) max = temp;
  }
```

```
void LastFour::update() {
    for (int i=3; i>0; --i) lastObserved[i] =
lastObserved[i-1];
    lastObserved[0] = subject->getLast();
  }
```

e) 2p

```
Temperature temp;
MinMax minmax;
temp.attach(&minmax);
minmax.setSubject(&temp);
LastFour lastfour;
lastfour.setSubject(&temp);
temp.attach(&lastfour);
for (i=1; i <= 36; ++i) temp.addTemp(i);
minmax.display();
lastfour.display();
```