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 B - 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 dea trei exemple de elemente comune si trei exemple de elemente ce diferențiază clasele abstracte de interfețe.
Răspuns.
0.5 p fiecare exemplu:
elemente comune: nu pot fi instanţiate, pot fi derivate/moştenite, includ definiţii de metode fără corp (doar numele şi signatura);
diferenţiere: clasele abstracte pot include şi implementări parţiale de metode, atribute, o clasă poate implementa/moşteni mai multe interfeţe dar numai o singură clasă abstractă, toate metodele dintr-o interfață sunt publice

2)

```
#include <iostream>
using namespace std;
class A{
public:
	A(int n ){cout << n;}
};
class X{
public:
	X(int n ){cout << n;}
};
class B: public A{
public:
	B(int n, double d): A(n)
	{cout << d;}
};
class C: public B, public X{
public:
	C(int n, double d, char ch): B(n,
d),X(++n)
	{cout <<ch;}
};
int main(){
	C c(5, 4.3, 'R');
	return 0;
}
```

Construiţi diagrama ierarhiei de clase din codul alăturat, descrieţi şi explicaţi ordinea de apel al constructorilor şi al destructorilor si precizaţi ce se afişează după execuţie şi cum este obţinut mesajul afişat (cine afişează fiecare componentă).
Răspuns.
- Diagrama 1p
- Ordinea de apel a cons/dest 1p
- rezultat **54.36R** 1p

3)

```
#include <iostream>
using namespace std;
template<class T>class clsTemplate{
public:
	T value;
	clsTemplate(T i=0){value = i;}
	void test(){cout << value;}
};
class clsChild : public clsTemplate<double>{
public:
	clsChild(): clsTemplate<double>( 1.1
){}
	clsChild(double x):
clsTemplate<double>( ){}

};
int main(){
```

În contextul programării generice, precizați care sunt definițiile parametrizate din codul alăturat și cum suntele utilizate. Care este rezultatul execuţiei programului și cum este obținut?
Răspuns.
- clasa parametrizata clsTemplate cu parametrul T (clasa) 1p
- utilizarea clasei parametrizate pentru definirea clasei clsChild 0.5p
- instanțierea claselor 0.5p
- rezultatul corect(A1.10) 1p

```
        clsTemplate<char> a('A');
        clsChild b, c(1.1);
        a.test();b.test();c.test();
        return 0;
}
```

4)

```
#include <iostream>
using namespace std;
class A{public:virtual int f() = 0;};
class B:public A{
public:
  int f(){cout << "B::f";return b;}
  int get_b(){return b;}
protected:
  static int b;
};
class C:public B{
public:
  C(){b++;}
  int f(){ cout << "C::f";return b;}
};
int B::b = 6;
int main() {
  A* a = new C;  a ->f();
  B* b = new C;  b ->f();
  cout <<(*b).get_b();
  delete a;delete b;return 0;
}
```

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

Răspuns.

C::fC::f8

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

5)

```
#include <iostream>
#include <queue>
using namespace std;
int main()
{
  int data[] = {12,21,13,31,15,51};
  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;
}
```

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.

13 4

13 31 15 12 0

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

6) (12 puncte)

Clasa *Book* este o de fapt o interfață care include elementele comune ale cărților: autor, titlu (ambele șiruri de caractere), preț (număr rațional). Clasa *Manual* este reponsabilă cu cărțile care sunt manuale. Un manual are în plus memorat nivelul (număr întreg, de ex. clasa). Clasa *Fiction* este resposabilă cu cărțile de ficțiune (beletristică), care au în plus categoria (șir de caractere).

a) Să se deseneze diagrama pentru șablonul *Visitor* și să se descrie pe scurt rolul fiecărui participant.

b) Să se intanțieze șablonul *Visitor* pentru a crea vizitatori care pot aplica discount pentru cărți și afișează informații specifice despre cărți, respectiv. Se instanțiază diagrama cu menționarea ce metode includ clasele din ierarhiile *Book* și *Visitor*.

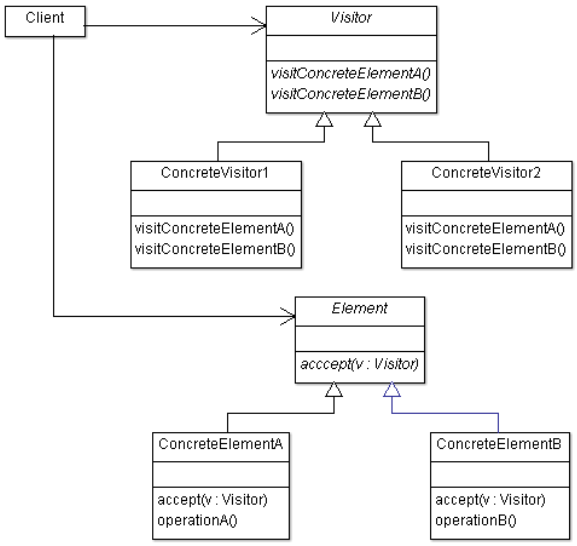
c) Să se scrie codul C++ al metodei de vizitare ce aplică discount pentru clasa Fiction. Discount-ul este 10% pentru cărți de ficțiune cu prețul <= 50 și 30% pentru cărți de ficțiune cu prețul > 50.

d) Să se scrie codul C++ al metodei de vizitare ce afișează informații despre manuale.

e) Să se scrie codul C++ care creează un tablou ce include 2-3 manuale și 2-3 cărți de ficțiune, creează un vizitator de discount și unul de afișare, apoi aplică cei doi vizitatori pentru obiectele din tablou.

Răspuns .

a) 2p



b)

```
class Book {
protected: double price;
protected: string author, title;
public: void init (string anAuthor = "", string aTitle = "", double aPrice = 0.0);
public: virtual void accept(Visitor&) = 0;
public: double getPrice();
public: string getTitle();
public: void setPrice(double newPrice);
};

class Manual : public Book {
private: int level;
public: Manual(string anAuthor, string aTitle, double aPrice, int aLevel);
public: void accept(Visitor&);
public: int getLevel() ;
};

class Fiction : public Book {
private: string category;
public: Fiction(string anAuthor, string aTitle, double aPrice, string aCategory);
public: void accept(Visitor& v);
public: string getCategory() { return category; }
};

class Visitor {
public: virtual void visitManual(Manual& man) = 0;
public: virtual void visitFiction(Fiction& man) = 0;
};

class Discount : public Visitor {
public: void visitManual(Manual& man);
public: void visitFiction(Fiction& fict);
};

class Display : public Visitor {
public: void visitManual(Manual& man);
public: void visitFiction(Fiction& fict);
```

```
};
```

c) 2p

```
void Fiction::visitFiction(Fiction& fict) {
  double disc;
  if (fict.getPrice() > 50) disc = 0.3;
  else disc = 0.1;
  fict.setPrice((1.0 - disc)*fict.getPrice());
  }
```

d) 2p

```
void Manual::visitManual(Manual& man) {
  cout << man.getTitle() << " " << man.getLevel() << " " << man.getPrice() << endl;
}
```

e) 4p (2p tabloul, 1p pentru fiecare vizitator)

```
    Book *books[20];
    int i, n = 5;
    books[0] = new Manual("Cauchy", "Analysis", 20, 11);
    books[1] = new Fiction("J.K. Rowling", "Harry Potter", 100, "children");
    ...
    Discount disc;
    for (i=0; i < n; ++i) books[i]->accept(disc);
    Display print;
    for (i=0; i < n; ++i) books[i]->accept(print);
```