Modele de bilete de examen Programare Orientata Obiect (POO) ASE

Programare Orientată Obiect Object-Oriented Programming (Academia de Studii Economice din Bucure□ti)

SUBIECT

(3p) Se consideră o aplicate pentru gestionarea activității unei magazin care vinde **baterii externe de tipul powerBank**. Se vor urmări atribute specifice, precum: capacitate greutate număr intrări/ieșiri, culoare etc. Datele membre sunt private si sunt puse la dispoziție metode de acces. Clasa conține **cel puțin patru câmpuri** dintre care **unul este alocat dinamic, constructori, metodele specifice** claselor cu **membri alocați dinamic** și **operatorul de afișare**. Folosiți un membru **static** sau **const**

(1p) Se vor defini **operator+=** si **operator-=** care permit incarcarea powerbank-ului respectiv descarcarea acestuia

(1p) Definiti **operator==** care compară doua obiecte de tip PowerBlank si returneaza true dacă toate valorile atributelor corespunzatoare sunt egale intre ele.

(2p) Exemplificati conceptul de relatie de tip „is a" prin specializarea clasei PowerBank. Testați soluția prin instanțierea noii clase.

(2p) Explicati conceptele de **early binding** si **late binding**.

(1p) Exemplificați conceptul de **functie template** in C++

SUBIECT

(2p) Se considera o aplicație pentru gestiunea unei colecţii de markere folosite pentru scrierea pe o tablă dintr-o sală. Se vor urmări aspectele comune privind culoare, dimensiune, producator, etc. Pentru datele membre private sunt puse la dispoziție metode de acces. Clasa **conține cel puțin patru câmpuri**, dintre care **unul este alocat dinamic**, **constructori, metodele specifice** claselor cu **membri alocaţi dinamic**. Folosiți un membru **static** sau **const**.

(1p) Supraîncărcați **operator<<** pentru afișarea unui marker.

(2p) **Specializați clasa** care descrie un **marker electronic** având noi câmpuri precum baterie, rază acţiune, etc.

(1p) Oferiți posibilitatea de comparare a două markere prin **operator==**, compararea realizându-se pentru minim două atribute.

(2p) Exemplificați conceptul de virtualizare prin oferirea unei funcționalități diferite pentru **Marker** şi **MarkerElectronic**.

(2p) Propuneți o metodă pentru a ține informaţiile despre marker-ele dintr-o sală. Dorindu-se ca pentru fiecare marker sa fie reţinut şi proprietarul acestuia. Un marker poate să aiba un singur proprietar, însă un proprietar poate să aiba mai multe markere. Propunerea realizata trebuie să permită identificarea proprietarului foarte uşor după marker.

SUBIECT

(3p) Se considera o aplicatie pentru gestiunea cartilor de vizita primite de catre o persoana Se vor urmän aspectele comune privind numele numanul de telefon, adresa de email, nume. companie, etc. Definiti o clasa care modeleaza un aspect propriu acestei activitati Se vor urmani atribute specifice, precum tipul materialului, numarul de exemplare, dimensiunile categorii, costuri etc. Datele membre sunt private si sunt puse la dispozitie metode de acces

Clasa contine **cel putin patru campuri** dintre care **unul este alocat dinamic**, **constructori, metodele specifice** claselor cu **membri alocati dinamic** si **operatorul de afisare** Folositi un **membru static**.

(1p) Toate cartile de vizita pot fi comparate cu **operator>=** comparatia este realizată dupa un atribut la alegere

(2p) **Specializați clasa** care descrie pentru un a carte de vizita in format electronic

(1p) Supraincarcati operator care permite citirea informatilar despre o carte de vizita de la tastatura

(2p) Exemplificati conceptul de virtualizare si late binding pun oferinea unei functionalitate diferite pentru cele doua clase

(1p) Propuneti o metoda pentru a gestiona cartile de vizita, astfel incat invegistranle sa se faca in mod unic sa nu exiite doua carti de vizita ale aceleasi persciane cu acelasi nume

Punctul din oficiu este inclus in prima cerinta. Neimplementarea acesteia ve conduce la notarea examenului 1

Pentru a luate in considerar solutile trebuie sa nu contina erori de compilare. Implementarea solutiei trebuie să te insotita de descrierea conceptelor folosite

SUBIECT

ProdusDerivat

(2p) Definiţi o clasă **abstractă Produs** care conține două metode:

- **calcul valoare** prin produsul dintre cantitate şi pret,
- respectiv **calcul TVA** pe bază de procent din valoare.

Clasa abstractă declară și datele necesare implementării acestor metode.

(1p) **Derivaţi** din ea o clasă care să modeleze comerțul electronic cu produse de încălțăminte şi instanțiați obiecte.

(1p) Supraîncărcați **operator<<** pentru afișarea informațiilor moştenite și specifice.

(2p) Definiti metode sau supraîncărcați operatori pentru creșterea / scăderea stocului prin aprovizionare / vânzare de produse.

(1p) Testați comportamentul virtual al metodei de calcul TVA pentru două categorii diferite de produse și servicii.

(1p) Comparaţi calculul TVA folosind **funcții virtuale** cu varianta în care fiecare produs are un **membru ce conține procentul de TVA** care i se aplică. Arătaţi avantajele și dezavantajele luând în considerare spațiul de memorie, volumul actualizărilor la schimbarea legislației, necesitatea recompilării claselor etc.

(2p) Adăugaţi în clasă două metode statice pentru **salvarea**, respectiv **restaurarea**, unei colecții de obiecte de tipul clasei (vector static) în/din **fişiere binare**

SUBIECT

(3p) Se consideră o aplicatie pentru gestionarea activităţii unui furnizor de energie electrică alternativă. Se vor urmări atribute specifice. precum: nume/denumire client, sursă energie, consum lunar efeleiv, consum lunar estimat, numar contract durată contract, pret kWh etc. Datele membre sunt private şi sunt puse la dispoziţie metode de acces. Clasa contine cel putin patru cămpuri, dintre care unul este alocat dinamic. constructori, metodele specifice claselor cu membri alocați dinamic şi operatorul de afișare. Pentru sursele de energie utilizaţi constante enumerative (enum). Folositi un membru static sau const.

(1p) Se va defini **operatorul funcție ()** pentru a modifica valoarea consumului lunar efectiv dintr-o anumită lună din contract.

(1p) Definiti **operatorul de conversie la double**, care va returna valoarea totala a diferentelor lunare de consum.

(2p) Scrieti **două metode**: o metodă pentru determinarea lunii cu cea mai mică diferență de consum şi o metodă care calculează valoarea totală a consumului efectiv.

(2p) Exemplificati **conceptul de virtualizare** prin utilizarea unei clase abstracte.

(1p) Propuneţi un **container STL** care permite efectuarea rapidă a operaţiilor de regăsire după numărul contractului.

SUBIECT

(3p) Se consideră o aplicație pentru gestionarea activităţii unei magazin care vinde baterii externe. Se vor urmări atribute specifice, precum: capacitate, nivel curent, producător, greutate, număr intrări/iesiri, tipuri intrări/ieşiri etc. Datele membre sunt private și sunt puse la dispoziţie metode de acces. Clasa conţine **cel puțin patru câmpuri**, dintre care **unul este alocat dinamic**, **constructori**, **metodele specifice claselor cu membri alocați dinamic** şi **operatorul de afişare**. Folositi un membru **static** sau **const**.

(1p) Se vor defini **operator+=** şi **operator-=** care permit încărcarea bateriei externe, respectiv descărcarea acesteia.

(1 p) Definiți **operatorul ==** care compară două obiecte de tip baterie externă şi returnează true dacă toate valorile atributelor sunt egale între ele.

(2p) Exemplificați conceptul de relatie de tip „is a" prin specializarea clasei de tip baterie externă. Testaţi soluția prin instanţierea noii clase.

(2p) Explicaţi conceptele de **early binding** şi **late binding**.

(1p) Exemplificați conceptul de funcție template în C++.

SUBIECT

(3p) Se consideră o aplicație pentru gestionarea activităţii unei **firme distribuție**. Definiţi o clasă care modelează un aspect specific acestei activităţi. Se vor urmări atribute precum: numele/denumirea clientului, produse şi cantități comandate, prețuri asociate etc. Datele membre sunt private și sunt puse la dispoziţie metode de acces. Clasa conţine cel puțin patru câmpuri, dintre care unul este alocat dinamic, constructori, metodele specifice claselor cu membri alocați dinamic şi operatorul de afişare. Folosiţi un **membru static sau const**.

(1p) Se va defini **operatorul[]** pentru a accesa un produs după poziţia dată.

(1 p) Definiţi **operatorul +=** pentru adăugarea unui produs la o comandă. Dacă produsul există deja in comanda, se va incrementa doar cantitatea.

(2p) **Scrieți într-un fişier text** comenzile care îndeplinesc un criteriu dat, primit ca parametru.

(2p) Exemplificați **conceptul de virtualizare** prin utilizarea unei clase abstracte.

(1p) Propuneți un **container STL** care permite efectuarea rapidă a operațiilor de inserare/stergere de produse intr-o/dintr-o comandă.

SUBIECT

(3p) Se considera o aplicație pentru gestiunea unei colecţii de **markere electronice** folosite pentru scrierea pe o tablă dintr-o sală. Se vor urmări aspectele comune privind culoare, dimensiune, producator, nivel curent acumulator etc. Definiți o clasă care modelează un aspect propriu acestei activități. Datele membre sunt private şi sunt puse la dispoziție metode de acces. Clasa conține cel puțin **patru câmpuri**, dintre care **unul este alocat dinamic**, **constructori**, **metodele specifice** claselor cu **membri alocați dinamic**. Folosiți un membru static sau const.

(1p) Supraîncărcați **operatorii <<** și **>>** pentru afișarea, respectiv citirea unui marker electronic.

(2p) **Specializați clasa** care descrie un marker inteligent având **noi câmpuri** precum: grosime, tip linie, nivel presiune etc.

(1p) Oferiți posibilitatea de comparare a două markere prin **operator= =**, compararea realizându-se pentru minim două atribute.

(2p) Exemplificați **conceptul de virtualizare** prin oferirea unei funcţionalități diferite pentru Marker şi MarkerElectronic.

(2p) Propuneţi o metodă pentru a ține informațiile despre marker-ele dintr-o sală. Dorindu-se ca pentru fiecare marker să fie reţinut şi proprietarul acestuia. Un marker poate să aibă un singur proprietar, însă un proprietar poate să aiba mai multe markere. Propunerea realizată trebuie să permită identificarea proprietarului foarte uşor după marker.

SUBIECT

(3p) Se consideră o aplicație folosita pentru a gestiona rezervarea unui sejur de vacanţă. Se vor urmari atribute specifice, precum: destinația, data de inceput, data de final, numarul de persoane, tip mese, tip camera, însoțitori etc. Datele membre sunt private sau protected. Clasa conține cel puțin trei câmpuri, dintre care unul este alocat dinamic, un constructor cu parametri, destructor, metode accesor (get și set) pentru unul dintre atribute. Folosiți un membru constant. Metoda set() validează datele de intrare.

(1p) Sa se supraincarce **operatorul+** in forma valoare + obiect pentru a modificare valoarea unui atribut.

(1p) Să se supraincarce operatorul de indexare [] care permite accesul in mod citire și scriere la un element dintr-un atribut de tip vector.

(2p) Exemplificati conceptul de relatie de tip „has a" prin definirea unei clase noi care să gestioneze mai multe obiecte de tipul clasei anterioare. Implementați o metodă care să permita adaugarea unui obiect in colecție. Testați soluția in main().

(2p) Exemplificati conceptul de serializare şi deserializare a unui obiect prin scrierea valorilor acestuia într-un fişier binar.

(1p) Exemplificati conceptul de funcție template in C++.

Punctul din oficiu este inclus in prima cerinţă. Neimplementarea acesteia va conduce la notarea examenului cu 1. Pentru a fi luate in considerare, soluțiile trebuie să nu conţină erori de compilare.

Implementarea soluție trebuie să fie insoțită de descrierea conceptelor folosite.

SUBIECT

SerieStatistica

(2p) Definiti clase care modelează lucrul cu o serie statistică sub forma unui vector dinamic de perechi valoare-frecvență, sortat după valoare.

(2p) Furnizaţi metode pentru calculul unor indicatori statistici (medie, dispersie, coeficient de corelație).

(2p) Supraîncărcați operator+ = pentru a obține o serie agregată a două serii statistice.

(1p) Supraîncărcați operator+= pentru a adauga o nouă pereche (valoare-frecvență) la o serie statistică existentă, menținând caracterul ei sortat.

(1p) Supraîncărcați operator-= pentru a elimina o pereche (valoare-frecvență) identificată prin valoare, dintr-o serie statistică existentă.

(2p) Transformați una din clase într-o clasă template sau instanțiați o clasă template STL care să faciliteze lucru cu serii statistice de forma unui vector de perechi valoare-frecventă. Indicaţi cum operează metodele elaborate mai sus în contextul clasei template.

SUBIECT

(3p) Definiţi o clasă care să permită gestiunea datelor aferente unui subiect de examen (titlu, barem pentru fiecare cerinţă, data examenului, etc.), folosind membri de tip public, private, protected, const, static Clasa contine un câmp alocat dinamic de perechi (cerinta, punctaj), constructori si două metode accesor (set va valida valoarea primita) pentru un atribut la alegere.

(1p) Supraincărcați operator+ care să permită implementarea operației string + obiect pentru completare enunt.

(2p) Supraincărcați operator<< pentru a permite scrierea subiectului într-un fişier de tip text. Testați in main pe un fişier text creat local.

(2p) Să se exemplifice utilizarea unei relații de tip has a de tip 1:1 (unu la unu) prin definirea unei clase suplimentare care să gestioneze evaluarea unui student. Pentru o clasa se defineste constructorul de copiere.

(1p) Exemplificați relația dintre constructorii celor două clase, aflate in relatia de 1:1, in situatia in care clasa inițială nu are constructor implicit.

(1p) Explicaţi şi exemplificati utilitatea pointerilor prin intermediul unui pointer la obiectele uneia dintre clasele definite anterior.