Programare orientată pe obiecte – Lista temelor pentru proiectul 2

Cerințe comune tuturor temelor (barem):

1. Toate clasele vor conține obligatoriu constructori de inițializare (0.25p), parametrizați (0.25p) și de copiere (0.25p); destructor (0.25p); operatorii „=" (0.25p), „>>" (0.25p), „<<" (0.25p) supraîncărcați corespunzător, moșteniri & encapsulare (0.5p)
2. Clasele derivate trebuie sa contina constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza) si destructori. (0.75p)
3. În fiecare proiect vor fi ilustrate conceptele de upcasting, downcasting, funcții virtuale (pure – unde se consideră mai natural) – 1.5p (0.5p / cerință)
4. Utilizarea de variabile și de funcții statice – 0.5p
5. Tratarea exceptiilor (***try – catch***) – 1p
6. Citirea informațiilor complete a n obiecte (***de diferite tipuri***), memorarea și afișarea acestora – 0.5p
7. Meniu interactiv – 0.5p
8. Rezolvarea corectă a cerințelor suplimentare corespunzatoare fiecarei teme – 2p.

Se acordă punctaje parțiale corespunzător și 1p oficiu.

- Dacă sursa nu compilează, se acorda nota 1 (punctul din oficiu).
- După expirarea termenului limită se mai pot trimite teme pe mail pentru o perioadă de grație de 2 zile (48 de ore). Pentru fiecare zi parțială de întârziere se vor scădea 2 puncte. După expirarea termenului de grație, proiectele nu vor fi acceptate și vor fi notate cu 0. Punctul din oficiu este primit doar dacă a fost trimis proiectul în perioada limită + perioada de grație.

Tema 1. Liste de numere întregi 1 (implementate dinamic)
Se dau următoarele clase:
-Nod (int info, Nod * next);
-Nod_dublu (Nod * ante) : Nod;
-LDI (elemente de tip Nod_dublu); // listă dublu înlănțuită
-LSI : LDI; //listă simplu înlănțuită, obținută mostenind caracteristicile unei LDI adaptate la o înlănțuire simplă;
Să se exemplifice sortarea prin inserție directă utilizând LDI.

Tema 2. Liste de numere întregi 2 (implementate dinamic)
Se dau următoarele clase:
-Nod (int info, Nod * next);
-Nod_dublu (Nod * ante) : Nod;
-Listă (elemente de tip Nod_dublu); // listă dublu înlănțuită
-Listă_Circulară : Listă;
Să se rezolve problema lui Josephus folosind liste circulare.

Tema 3. Departamentul „Salarizare" al firmei X actualizează fișele întregului personal utilizând o aplicație OOP. În acest scop se vor implementa clasele:
-Data (int zi, char *luna, int an);
-Angajat (string nume, string prenume, float salariu, Data data_angajare);

-Part_Time (int nr_ore_zi, Data final_contract) : Angajat
-Permanent (int nr_minori_intretinere) :Angajat
De sărbători fiecare angajat va primi o primă. Angajatul permanent va primi pe lângă suma standard și un spor calculat ca un procent din prima fixă, egal cu vechimea lui în firmă, calculată până la 31 decembrie 2020 (de exemplu: dacă vechimea lui în firmă este de 12 ani, atunci va primi pentru pentru fiecare copil minor 12% * suma standard stabilită de firmă, etc), iar angajații part-time vor primi suma standard stabilită dacă contractul lor nu se termină până la sfârșitul lunii curente, altfel, primesc doar 75% din suma standard prevăzută.

Tema 4. Se dau urmatoarele clase:
- Clasa Persoana(int id, string nume)
- Clasa Abonat:Persoana(string nr_telefon)
- Clasa Abonat_Skype: Abonat (string id_skype)
- Clasa Abonat_Skype_Romania (string adresa_mail) : Abonat_Skype
- Clasa Abonat_Skype_Extern(string tara) : Abonat_Skype

Sa se construiasca clasa Agenda ce contina o lista de abonati si sa se supraincarce operatorul [ ](indexare)care returneaza abonatul cu numele precizat.

Tema 5. Se dau urmatoarele clase:
- Punct (float x, float y)
- Patrat (Punct stanga_jos, float latura)
- Dreptunghi (float latura2) : Patrat
- Romb (Punct colt opus) :Patrat
- Paralelogram : Dreptunghi, Romb
- Trapez (float baza2) : Paralelogram

Toate figurile au 2 laturi paralele cu axa Ox. Clasele derivate trebuie sa contina constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza) si destructori. Functiile membre conțin și metode de calcul pentru arie și volum.
O data membra **valid** are valorea 1 dacă figura este specificata corect.
Constructorii verifica paralelismul laturilor.
Definiti și implementati ierarhia de clase.

Tema 6. Se dau următoarele clase:
-Abonament (string nume_abonament, float pret, int perioadă)
-Abonament_Premium (int reducere) : Abonament
-Persoană (int id, string nume, string cnp)
-Abonat (string nr_telefon, abonament x) : Persoana
Să se construiască clasa Clienți care reține o listă de abonați. Să se afle numărul de abonați premium. Să se realizeze o metodă care află care este suma de bani încasată de la toți abonații considerând perioada ca fiind nr de luni si prețul este plătit pe fiecare lună.

Tema 7. Se dau clasele:
-Locuință (string numeClient, int suprafataUtila, float discount)
-Apartament (int etaj) : Locuinta
-Casa (int suprafataCurte) : Locuinta

La clasa Locuinta se va adauga metoda virtuala pura CalculChirie (X,Y) cu X = valoare standard chirie/mp(intreg), Y=1 daca se ia in considerare discountul si 0 daca nu se ia in considerare.
Metoda va fi adaugata in clasa Apartament dupa formula X*suprafataUtila*(1-Y*discount/100.0), respectiv in clasa Casa dupa formula X*(suprafataUtila + 0.2*suprafataCurte) * (1-Y*discount/100.0).
Metodele vor fi testate prin parcurgerea unui vector de pointeri la Locuinta *, incarcat cu obiecte de tip Apartament si Casa.
Se defineste clasa AgentieImobiliara continand un vector de pointeri la obiecte de tip Locuinta alocat dinamic. Se va supraincarca operatorul >> pentru a citi locuintele agentiei si operatorul << pentru afisarea lor.

Tema 8. Să se definească clasa Contract care conține membrii privați

- nrContract (numar contract)
- anul (anul semnarii contractului de cumparare)
- beneficiar (numele cumparatorului/beneficiarului)
- furnizor (nume vanzator/ofertant/furnizor)
- valoare (valoarea totala a produsului)

Să se deriveze clasa ContractInchiriere din Contract având în plus ca membru privat

- perioada (perioada contractului exprimata in numar luni)

Definiti clasa Dosar care contine un vector de pointeri la obiecte de tip ContractInchiriere si un numar de contracte de inchiriere.
Se citesc contractele din dosar, se afiseaza si se cere sa se calculeze valoarea incasata pentru fiecare contract in functie de perioada, respectiv valoarea totala incasata.

Tema 9. Se dau clasele:
-Avocat (int cod, string nume, int nr_procese, int *vector_procese)
-Avocat_Oficiu (string nume_client, int durata) : Avocat.
Vectorul de procese retine suma primita pentru fiecare proces.
Să se supraincarce operatorii de comparare corespunzator astfel incat sa se sorteze o lista de avocati dupa numarul de procese si definiti o metoda care gaseste dintr-o lista de avocati din oficiul avocatul care a petrecut timpul maxim cu un client.

Tema 10. matrice de numere complexe reprezentate ca tablouri
bidimensionale
Se dau urmatoarele clase:
- Clasa Complex(float re,im)
- Matrice(Complex **v)
- Matrice_oarecare (int lin, int col) : Matrice
- Matrice_patratica (int dim): Matrice

Scrieti o metoda care sa verifice daca o matrice triunghiulara este diagonala. Pentru matricile patratice, functia de afisare sa conțină șideterminantul acestora.

Tema 11. Se dau clasele:

- Persoana (string nume, string cnp);
- Abonat (int nrMaxCarti, int nrCartiImprumutate, int pretAbonament) : Persoana

Sa se creeze clasa dosar care retine informatiile abonatilor la o biblioteca si contine un vector de pointeri la obiect abonat si lungimea acestuia.

Sa se realizeze verificarile necesare in constructor si la citire astfel ca sa nu poti avea mai multe carti imprumutate decat numarul maxim de carti si numarul maxim de carti sa nu poata fi mai mare decat pretul abonamentului. Sa se defineasca o metoda de calculare a varstei persoanei din cnp, sa se citeasca si afiseze n obiecte de tip abonat care sa fie adaugate in dosar.

Tema 12. Se dau clasele:

- Card (string nrCard, string NumeDetinator, string data_expirare, int CVV, double credit) [cvv=cele 3 cifre de pe spatele cardului)
- Card_standard (int limitaExtragere, double comisionDepasireLimita) : Card
- Card premium (double cashback) : Card_Standard

Sa permiteti printr-o metoda sa extrageti bani de pe un card. Sa se verifice ca aceasta suma sa nu fie mai mare decat creditul. Sa se modifice creditul la extragere in functie de card standard sau premium. Daca depaseste limita cardului standard atunci se aplica comisionul pentru diferenta de bani. Daca se extrage de pe un card premium sa primeasca inapoi cashback/100 * suma retrasa.

Tema 13. polinoame reprezentate ca tablouri unidimensionale (prin gradul polinomului si vectorul coeficientilor).
Se dau urmatoarele clase:
- Clasa Monom(int grad, float coef)
- Clasa Polinom(int nr_monoame, Monom *m)
- Polinom_ireductibil : Polinom;
- Polinom_reductibil : Polinom.

Clasele derivate trebuie sa contina constructoi parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza), destructor si o metoda care sa aplice criteriul lui Eisenstein de verificare a ireductibilitatii polinoamelor.
(Webografie ajutatoare: **http://www.profesoronline.ro/teorie-3140-1.html)**
Afisarea unui polinom reductibil să fie făcută și ca produs de 2 polinoame.

Tema 14. Se dau clasele:

- Proces (int nrProces, string reclamant, string reclamat)
- Proces_civil (double dauneMorale, double dauneMateriale, int nrMartori, bool stadiu)
- Proces_penal (int dovezi, bool stadiu).

Sa se faca verificarile in constructori si la citire astfel: daca nrMartori > 5 automat stadiul este 1, altfel este 0. Daca la un proces penal numarul dovezilor > 25 atunci stadiul este 1.
Sa se poata modifica stadiul unui proces si sa se creeze o metoda de calculare a taxei de timbru pentru fiecare proces civil. Taxa de timbru = 10/100 * dauneMorale + 10% * dauneMateriale.
Sa se afle procesul care are taxa de timbru cea mai mare.