LUCRAREA PRACTICĂ NR. 3

Precizări generale

- Temele sunt individuale. Fiecare student își asumă responsabilitatea realizării unei teme. Modul de notare aplicabil tuturor temelor este descris în barem.
- Pentru orice porțiune de cod inclusă din alte surse, sursa trebuie citată în cod prin inserarea unui comentariu (aici prin surse înțelegem cărți/curs/seminar/tutoriat/internet). Nu este permisă preluarea din alte surse a mai mult de 3-4 linii de cod (orientativ).
- Este încurajată solicitarea și acordarea de ajutor punctual de la colegi vis-a-vis de o anumită chestiune specifică (e.g. “ai mai întâlnit eroarea X?”).
- Este, în același timp, interzisă înglobarea de cod scris de colegi sau scrierea după dictare a codului, ori preluarea integrală a codului aferent temei din terțe surse. Se consideră fraudă și se vor aplica consecințele prevăzute în regulamentul prezent pe Teams, în secțiunea Files.
- Cele mai multe cerințe sunt parțiale, lasând la latitudinea voastră anumite elemente concrete pentru a ilustra cât mai bine competențele voastre de POO.
- În cazul în care **o cerință este neclară** îmi puteți trimite un e-mail cu nelămurirea pe adresa paul.sumedrea@s.unibuc.ro sau un mesaj în chatul privat din Teams.
- Pentru feedback, vom folosi exclusiv Github (feedback pentru cod: https://classroom.github.com/a/IJ011tm8) și orele de laborator alocate (pentru întrebări legate de conceptele necesare)
- Studenților care nu sunt prezenți în cadrul laboratorului pentru a își alege o temă li se va atribui una din oficiu.

  Termen limită pentru predarea LP3:

  16 mai, ora 23:59 (fără penalizări)

  18 mai, ora 23:59 (cu penalizări; -1p per zi de întârziere; după această dată orice temă trimisă va fi notată cu 0p)

Temele vor fi trimise prin e-mail la adresa: **paul.sumedrea@s.unibuc.ro**

Titlu mail: **NrProiect_Nume_Prenume_Grupa_LP3_FINAL**

E-mail-ul va conține o singură arhivă, în format .zip. Numele arhivei va fi de forma: **NrProiect_Nume_Prenume_Grupa_LP3_Final.zip**

Arhiva va conține fișierele **NrProiect_Nume_Prenume_Grupa_LP3.cpp** (vă rog să redenumiți main.cpp după forma aceasta), **README ul** (în format **.pdf**) și, eventual, alte fișiere .h sau .cpp + fișierele de intrare-ieșire pe care le folosiți în proiect.

NU INCLUDEȚI în arhivă întreg proiectul de CodeBlocks/Clion/VSCode etc.

Cerințe comune tuturor temelor (BAREM)

1. Identificarea și implementarea ierarhiei de clase (având grijă la definirea exhaustivă a variabilelor necesare și la tipul metodelor folosite pentru rezolvare) **1p**
Toate clasele vor conține în mod obligatoriu:

- constructori de inițializare (**0.1p**);
- constructori parametrizați (**0.15p**);
- constructori de copiere (**0.15p**);
- destructori (**0.15p**);
- operator„=” (**0.25p**),
- operatorii „>>” (**0.25p**), „<<” (**0.25p**) supraîncărcați corespunzător sau funcții virtuale echivalente pentru rezolvarea acestor funcționalități

2. Clasa de bază va conține funcții virtuale de afișare și citire, rescrise în clasele derivate, iar operatorul de citire și afișare va fi implementat ca funcție prieten (în clasele derivate să va face referire la implementarea acestuia în clasa de bază) **0.2p**

3. utilizarea și definirea corectă a șabloanelor **1p**

4. utilizarea STL **1p**

5. tratarea excepțiilor (cu try catch; minim 1 - maxim 2 excepții verificate per variabilă, + se vor depuncta toate erorile logice care nu sunt tratate) **0.5p**

6. În fiecare proiect vor fi ilustrate conceptele de RTTI raportat la template-uri (ex: upcast & dynamic_cast) **1p**

7. Utilizarea variabilelor și a funcțiilor statice, a constantelor și a funcțiilor const – **0.5p**

8. Eliberarea memoriei - **0.5p**

9. Rezolvarea corectă a cerinței marcate în dreptul fiecărei teme **0.5p**.

10. Citirea informațiilor complete a N obiecte (alocate dinamic), memorarea și afișarea acestora într-un meniu interactiv (cu demonstrarea funcționalităților în README prin capturi de ecran/screen recording, la alegere) – **0.5p**

11. Prezentarea codului (prin README/comentarii) și răspunsul la întrebări - **0.5p**

- Se vor acorda punctaje parțiale corespunzător și respectiv **1p** oficiu.

- Dacă sursa **NU COMPILEAZĂ**, se acorda nota 0.

Data prezentării este: 21.05.2021

Lipsa/abaterea de la convențiile amintite în acest document legat de formatul e-mailul ce conține soluția finală va atrage o depunctare de **0.5 puncte** din nota aferentă LP3!

Lista temelor, cu toate cerințele pentru a treia lucrare practică.

Tema 1.

Firma X are un domeniu de business unde este necesar să se urmărească modul în care clienții plătesc (numerar, cec sau card de credit). Indiferent de modul de plată, firma X știe în ce data s-a efectuat plata și ce sumă a fost primită. Dacă se plătește cu cardul, atunci se cunoaște și numărul cardului de credit. Pentru cash, nu este necesară identificarea clientului care a făcut plata.

Structura de date: unordered_map sau unordered_set <id_plata, structura care reține datele plății>

Să se construiască clasa template Gestiune care sa conțină numărul total de plăți de un anumit tip (incrementat automat la adaugarea unei noi chitanțe) și structura de obiecte de tipul plății, alocată dinamic. Să se supraincarce operatorul += pentru inserarea unei plăți în vector.

Să se adauge toate câmpurile relevante pentru modelarea acestei probleme.

Cerința suplimentară:

- Să se construiască o specializare aferentă tipului de plata prin card, care să stocheze și numărul de clienți, împreună cu numele acestora. Specializarea va adapta operatorii menționați în cerințe și operatorul += la acest tip de plata.

Tema 2.

La ora de Biologie, copiii din ciclul gimnazial învață că regnul animal se împarte în 2 grupuri: nevertebrate și vertebrate. La rândul lor, vertebratele se împart în pești, păsări, mamifere și reptile.

Structura de date: list<Animal*>

Să se construiască clasa template AtlasZoologic care să conțină un număr de animale (incrementat automat la adăugarea unei noi file) și structura de obiecte de tipul

regnurilor implementate, alocată dinamic. Să se supraîncarce operatorul += pentru inserarea unei fișe de observație a unui animal în vector.

Să se adauge toate câmpurile relevante pentru modelarea acestei probleme.

Cerința suplimentară:

- Să se construiască o specializare pentru tipul Pești care să adapteze operatorii menționați și care să afișeze, în plus, câți pești răpitori de lungime mai mare de 1m s-au citit.

Tema 3.

La facultatea Y studenții intră în sesiune. În regulament este prevăzut ca ei să aibă un anumit număr de examene. Fiecare examen are un număr, incrementat automat, denumirea materiei, data și nota la scris. Parțialul conține și nota la oral, iar examenul final conține extra puncte primite pe un proiect. Dacă parțialul nu e promovat, atunci acesta se va reface la examenul final. Altfel, i se păstrează nota. Cei care vor să-și mărească nota, mai dau un quiz, conținând un număr de itemi de tip grilă.

Structura de date: unordered_set sau unordered_map <id_examen, vector<elev>> (se reține elevii care nu au trecut examenul cu id-ul id_examen)

Să se construiască clasa template CatalogIndividual care sa conțină informații despre tipurile de examene date de un student. Clasa conține nr matricol al studentului (incrementat automat la adăugarea unei noi file), numărul de examene și structura de obiecte pentru examene. Să se supraîncarce operatorul += pentru inserarea unei fișe de observație a unui examen în structură.

Să se adauge toate câmpurile relevante pentru modelarea acestei probleme.

Cerința suplimentară:

- Să se construiască o specializare pentru tipul Quiz care să adapteze operatorii menționați și care să afișeze, în plus, câte persoane au dat cel puțin 2 quiz-uri.

Tema 4.

Dintr-un parc auto se poate cumpăra o gamă variată de automobile de următoarele tipuri: MINI (mașină de oraș, de mic litraj, de obicei sub 4m lungime), MICA (mașină de

oraș, cu spațiu interior mai mare decât MINI și lungime între 3.85 și 4.1), COMPACTA (mașină ușor de folosit, atât în oraș, cât și la drum lung, de dimensiune 4.2 – 4.5m; acest tip de mașini are formă de hatchback, combi sau sedan) și MONOVOLUME (automobile sub formă de van, ce pot transporta 5-7 persoane). Monovolumele pot fi achiziționate atât noi cât și second hand. La cele achiziționate second hand, se percepe un discount proporțional cu numărul de ani vechime ai mașinii.

Pentru toate automobilele, în lunile de vară, se beneficiază de zile promoționale cu reducere fixă de 10% din preț.

Structura de date: set<pair<tip_automobil, bool nou>> (nou = false pentru cele sh)

Să se construiască clasa template Vanzare care sa conțină numărul total de mașini aflate pe stoc (decrementat automat la vânzarea unei mașini), numărul de mașini vândute (incrementat automat) și două structuri de obiecte, alocate dinamic, prin care să se gestioneze automobilele din stoc și cele vândute. Să se supraîncarce operatorul -= care să actualizeze datele din clasă la vanzarea unei mașini.

Să se adauge toate câmpurile relevante pentru modelarea acestei probleme.

Cerința suplimentară:

- Să se construiască o specializare pentru tipul MONOVOLUM care sa conțină și să afișeze/gestioneze doar MONOVOLUMELE.

Tema 5.

În luna mai, se organizează târgul mașinilor sport. Astfel, pasionații se pot delecta cu modele din clasa COUPE, HOT-HATCH (modele hatchback de clasa mică și compactă cu motoare performante ce au la bază modele obișnuite), CABRIO (modele decapotabile cu acoperiș metalic sau din material textil) și SUPERSPORT(mașini de viteza gen Audi R8, Bugatti Veyron, Lexus LFA, etc.). O parte din mașinile supersport pot fi vândute chiar în cadrul expoziției, dacă tranzacția se face cu plata pe loc.

Structura de date: vector sau list <pair<masina, preț>> (se reține mașinile vândute și prețul de vanzare, dacă mașina nu a fost vanduta prețul este -1)

Să se construiască clasa template Expoziție care să conțină numărul total de mașini expuse (incrementat automat) și un vector de obiecte de tipul unei mașini, alocat dinamic.

- Să se adauge toate campurile relevante pentru modelarea acestei probleme.

Cerința suplimentară:

- Să se construiască o specializare pentru tipul SUPERSPORT care sa conțină numărul total de mașini supersport expuse (decrementat automat la vanzarea unei mașini), numărul de mașini vândute (incrementat automat) și doi vectori de pointeri la obiecte de tip mașina supersport (două structuri alocate dinamic) prin care să se gestioneze automobilele supersport din stoc și automobilele supersport vândute. Să se supraîncarce operatorul -= care să actualizeze datele din clasă la vânzarea unei mașini.

Tema 6.

Se dorește implementarea unei aplicații OOP care să permită gestionarea activității unor farmacii aparținând proprietarului X. Pentru fiecare farmacie se cunosc cel puțin denumirea, numărul de angajați și profiturile farmaciei din fiecare lună. Farmaciile pot fi de asemenea și strict online. În acest caz se va cunoaște măcar adresa web, numărul de vizitatori și discountul utilizat.

Structura de date: vector sau list <tuple<web, nr_vizitatori, discount>> se rețin farmaciile online

- Să se construiască clasa template GestionareFarmacii care sa conțină informații despre diversele tipuri de farmacii. Clasa conține indexul farmaciei (incrementat automat la adaugarea unei noi file), id-ul lanțului (constant) și o structură de obiecte, alocată dinamic. Sa se supraîncarce operatorul += pentru inserarea unei noi farmacii online în structură.

- Să se adauge toate câmpurile relevante pentru modelarea acestei probleme

Cerința suplimentară:.

- Să se construiască o specializare pentru tipul Farmacie_online care sa conțină și să afișeze doar numărul total de vizitatori ai farmaciilor online.

Tema 7

Se dorește implementarea unei aplicații care să permită gestionarea conturilor deschise la banca X. Fiecare cont bancar are obligatoriu un deținător, o dată a deschiderii lui și

un sold. În cazul conturilor de economii, trebuie reținută și rata dobânzii (care poate fi pe 3 luni, pe 6 luni sau la un an), precum și un istoric al reactualizarii soldurilor de la deschidere și până în prezent.

În cazul în care deținătorul optează pentru un cont curent, el beneficiază de un număr de tranzacții gratuite și altele contra cost pe baza unui comision fix (de exemplu: orice depunere este gratuită, dar retragerea poate să coste dacă s-a depășit numărul de tranzacții gratuite sau este făcută de la bancomatele altor bănci; puteți considera de asemenea că orice tranzacție online de criptomonede va avea un comision, orice cumpărătură online are un comision suplimentar etc.). Simulați cât mai corect activitatea băncii X.

Structura de date: unordered_map sau unordered_set <id_cont, list sau vector <operațiuni pe contul id_cont>>

Să se construiască clasa template GestionareConturi care să conțină informații despre banca X. Clasa conține indexul unui cont (incrementat automat la adaugarea unui nou cont prin supraincarcarea operatorului +=) și o structură de date.

Să se adauge toate câmpurile relevante pentru modelarea acestei probleme.

Cerința suplimentară:

- Să se construiască o specializare pentru tipul Cont_Economii care să afișeze toate conturile de economii care au rata dobânzii la 1 an.

Tema 8

Pizzeria X testează un nou soft, dezvoltat în maniera OOP, pentru gestionarea activității sale. Produsele propriu-zise ale pizzeriei sunt de forma unei clasă abstracte ce conține doar o metodă virtuală pură de calcul a prețului unui produs. Orice pizza gătită de pizzeria X se consideră a fi un tip de produs (pizzele pot fi vegetariene, cu diferite branzeturi sau cu diferite tipuri de carne). În realizarea oricărei pizza intră un anumit număr de ingrediente, pentru care se cunosc denumirile, cantitățile și prețul unitar. Prețul unei pizza este dat de prețul ingredientelor, plus manopera (o sumă constantă, fixată de producător). Daca pizza este comandata OnLine, la prețul final se mai adaugă un comision de livrare de 5% din prețul pizzei, pentru fiecare 10 km parcurși de angajatul care livrează la domiciliu.

Structura de date: unordered_map sau unordered_set <id_pizza, list sau vector <ingredient>>

Să se construiască clasa template Meniu care sa gestioneze tipurie de pizza comercializate. Clasa trebuie să conțină indexul unui produs (incrementat automat la vanzarea unui produs, prin supraincarcarea operatorului +=) și o structură de date, alocată dinamic.

Să se adauge toate câmpurile relevante pentru modelarea acestei probleme.

Cerința suplimentară:

- Să se construiască o specializare pentru tipul Comanda_Online care să trateze tipurile de pizza vegetariană într-un document contabil separat, din care să rezulte valoarea totală încasată din vânzarea acestora.

Tema 9

Se dorește implementarea unei aplicații care să permită gestionarea clienților și a proprietăților imobiliare în cadrul unei agenții imobiliare nou înființate care închiriază proprietăți. Pentru fiecare locuință se cunoaște cel puțin numele clientului care o închiriază, suprafața utilă și discount-ul (0-10%). La apartamente se cunoaște etajul, iar pentru case se cunoaște câți metri patrati are curtea, câte etaje are casa și care este suprafața utilă pentru fiecare etaj în parte.

Evident, calculul chiriei se face diferit. Dacă la apartamente se consideră doar formula dată de prețul de închiriere pe metru pătrat * suprafața utilă (având grijă să se aplice discount unde este cazul) la casă se adaugă, indiferent de discount, prețul pe metru pătrat de curte * suprafața acesteia.

Structura de date: set<pair<locuinta, tip>>, unde tip = apartament sau casa

Să se construiască clasa template Gestiune, conținând structura de obiecte de tipul locuințelor implementate, alocat dinamic, unde indexul fiecărei locuințe este incrementat automat la adăugarea uneia noi, prin supraîncărcarea operatorului +=.

Să se adauge toate câmpurile relevante pentru modelarea acestei probleme.

Cerința suplimentară:

- Să se construiască o specializare pentru tipul Casa care să stocheze numărul de case, fiecare cu chiria aferentă și afișează totalul obținut de agentia imobiliara de pe urma acestora.

Tema 10

Sa se proiecteze o aplicație de gestiune a resurselor umane pentru un grup de companii. Se cunoaște faptul că toate companiile dețin o serie de departamente (ex: “derulare proiecte”, “financiar”, “IT”, ...) și au o serie de angajați, distribuiți în departamente, unii dintre aceștia putând avea funcția de manageri pentru alți angajați. Fiecare angajat poate avea cel mult un manager.

Gestionarea se va realiza prin intermediul unei clase template numite GestionarHR. Aplicația de gestiune ce se dorește a fi implementată trebuie să poată gestiona pe lângă numele angajatilor și informații cum ar fi data de angajare în companie și salariul lunar brut al fiecărui angajat. Se vor realiza interfețe prin care să se poată adăuga/șterge departamente dintr-o companie, angajați sau chiar și întregi companii din gestionarul de companii.

De asemenea, aplicația trebuie să poată furniza rapoarte legate de: numărul de angajați ai unei companii sau al unui departament dat; cheltuielile salariale lunare totale existente la nivelul unei companii; numele, data de angajare și salariul angajatilor care au funcția de manageri, precum și lista de angajați care le sunt subordonați acestora, la nivelul companiei din care fac parte; lista departamentelor care au minim N angajați (N transmis ca parametru).

Trebuie să existe posibilitatea de fuzionare a doua companii (se va defini un operator + care realizeaza acest lucru, precum și o metoda la nivelul gestionarului). Departamentele comune vor fuziona într-unul singur. La fuzionare toți managerii își păstrează funcția.

Aveți în vedere faptul că la adăugarea unei companii în gestionar trebuie să se verifice dacă acea companie nu există deja în gestionar. O companie este identică cu o alta dacă are același nume și aceeași schemă de departamente (același număr de departamente).

Structura de date: set<pair<nume_companie, nr_departamente>>

Să se adauge toate campurile relevante pentru modelarea acestei probleme.

Cerința suplimentară:

Pornind de la modelul proiectat mai sus, să se adauge posibilitatea de lucru cu companii care beneficiază de scutire de impozit (dețin departamente cu scutire de impozit (16%)). Acolo cheltuielile salariale sunt diminuate.

Tema 11.

La realizarea unui film (ce are cel puțin un nume un gen și o durata), participă o mulțime de persoane: un regizor, actori, personal tehnic etc. Dacă filmul este bun, atunci el rulează într-un număr de cinematografe și realizeaza încasări substanțiale pentru fiecare cinematograf în parte. Fiecare membru implicat în realizarea filmului (pentru care se cunoaște cel puțin cnp-ul, numele său și numele filmului/filmelor în care a jucat) are prevăzut prin contract un procent din încasări. Drept urmare acel membru primește, pentru fiecare film la care participa, o anumită sumă. În plus, regizorul are prevăzută și o sumă fixă pe care și-a negociat că o va primi, indiferent de calitatea filmului produs.

Să se construiască clasa template FirmaDistributie care sa conțină informații despre filmele realizate într-un an. Clasa conține numărul total de persoane implicat incrementat automat), numărul de actori (incrementat automat) și un vector de persoane alocat dinamic.

Structura de date: unordered_map sau unordered_set<nume_membru, list sau vector <filme>>

Să se adauge toate câmpurile relevante pentru modelarea acestei probleme.

Cerința suplimentară:

- Să se realizeze o specializare pentru tipul Actor care sa conțină numărul actorilor care au fost distribuiți în roluri principale. Acești actori vor primi pe lângă salariu și un bonus de 10% din încasări.

Tema 12

Agenția de turism Y oferă posibilitatea rezervarii unor sejururi în mai multe destinatii turistice (excursie la munte, excursie la mare, city-break). Indiferent de destinație, pentru a realiza o rezervare agenția de turism trebuie să cunoască perioada dorită pentru sejur (ca număr de zile) și destinația dorită. În baza acestora, agenția va oferi o estimare a prețului excursiei. Dacă destinatia este montană, atunci la prețul excursiei i se va adăuga valoarea inchirierii echipamentului de ski și se va cunoaște numele monitorului de ski care îi este atribuit turistului. Pentru city break se cunosc numele muzeelor ce pot fi vizitate (la care se poate face rezervare în prealabil) și prețul unui bilet la muzeu (care se adaugă la costul total). Pentru excursiile la mare, prețul excursiei

va include suplimentar și o taxă de șezlong ce va fi plătită în avans pentru fiecare zi de ședere la mare.

Se presupune ca fiecare destinație are un tarif standard asociat pentru sejur pentru fiecare zi de cazare.

Structura de date: unordered_map sau unordered_set <id_rezervare, structura care reține datele rezervarii>

- Să se adauge toate campurile relevante pentru modelarea acestei probleme.

- Să se construiască clasa template BookingVacante care sa conțină numărul total de rezervari de un anumit tip (incrementat automat la adaugarea unei noi rezervări) și structura de obiecte de tipul destinației, alocat dinamic. Sa se supraincarce operatorul += pentru inserarea unei rezervari în vector.

Cerința suplimentară:

- Să se construiască o specializare aferentă excursiilor la mare care să permită gestionarea excursiilor la mare și aflarea prețului total al excursiilor rezervate la mare de către agenție.

Tema 13

Se dorește implementarea unei aplicații OOP care să permită gestionarea rezervărilor la sălile de spectacol apartinand proprietarului Y. Pentru fiecare sală de spectacol se cunoaște cel puțin tipul spectacolului care se desfășoară (e.g. opera, teatru), numele spectacolului, genul acestuia (de ex: comedie, drama, etc.) și prețul biletului. Cunoaștem despre toate spectacolele cel puțin faptul că au o durata și un nume, durata fiind variabila. Piesele de teatru au un singur act. Unele piese de teatru sunt interactive (presupun participarea publicului). Despre spectacolele de operă se cunoaște faptul că au un solist principal. Solistul principal are un nume și poate fi tenor, bariton sau bass. Unele opere pot avea mai multe acte (2-7), cu durate diferite, specificate pentru fiecare act în parte. Între acte se ia o pauza fixă specifică operei ce se va adăuga la durata totală a piesei, pentru a putea să îi estimăm durata totală.

Să se construiască clasa template GestionareRezervariSpectacole care sa conțină informații despre diversele tipuri de spectacole și să permită efectuarea unor rezervări pentru acestea. Clasa conține indexul rezervarii (incrementat automat la adaugarea unei noi rezervări), și o structura de obiecte, alocată dinamic care sa gestioneze rezervările care se pot face la un anumit spectacol.

Structura de date: unordered_map sau unordered_set <id_rezervare, structura care reține datele spectacolului rezervat>

- Să se adauge toate câmpurile relevante pentru modelarea acestei probleme.

Cerința suplimentară:

Dat fiind că la sălile de spectacol au început să fie prezentate și spectacole de stand-up, să se construiască o specializare aferentă acestui tip de spectacol ce ne va permite să aflăm veniturile obținute de proprietarul Y pe baza biletelor vândute la stand-up.

Tema 14

Se dorește implementarea unei aplicații care sa permita gestionarea rezervărilor în cadrul hotelului Y. Hotelul Y are mai multe tipuri de încăperi (camere, apartamente și penthouse-uri). Pentru fiecare spațiu de cazare se cunoaște numele persoanei pe numele căreia s-a făcut rezervarea pentru respectivul spațiu, numărul de persoane cazate în spațiul respectiv și numărul de zile rezervate. Fiecare tip de spațiu de cazare are un număr de identificare unic în cadrul hotelului. Camerele simple au un număr cunoscut de scaune si paturi. Pentru apartamente se cunoaste si numarul de bai, iar pentru penthouse se cunoaste numărul de pisicine.
Pentru fiecare spațiu de cazare calculul prețului sejurului se face diferit. Dacă la camerele simple se consideră doar formula dată de rata standard de cazare per zi de practicata de hotel * numărul de persoane cazate în respectivul spațiu, pentru apartamente se adaugă 10% * nr de bai la acest calcul iar penthouse-urile sunt cu 1000 de lei mai scumpe per zi de cazare in functie de numărul de pisicine.

Să se construiască clasa template GestiuneRezervari, continand structura de obiecte de tipul spațiilor de cazare implementate, alocat dinamic, unde indexul fiecărui spațiu de cazare este incrementat automat la rezervarea unui nou spațiu, prin supraincarcarea operatorului +=.

Structura de date: set<pair<numar_spatiu_de_cazare, tip>>

- Să se adauge toate campurile relevante pentru modelarea acestei probleme.

Cerința suplimentară:

- Să se construiască o specializare pentru tipul Penthouse, care sa stocheze numărul de astfel de spații, împreuna cu prețul asociat și afișează totalul obținut de hotel de pe urma rezervarii acestora.

Tema 15

Să se realizeze o aplicație C++ care să permită ținerea evidenței articolelor prezente într-o biblioteca studențească (cărți, reviste și culegeri de probleme). Orice articol existent în bibliotecă va conține o cotă. Cota articolelor din biblioteca folosește la căutarea și împrumutarea acestora de către studenți. Fiecare carte are un titlu, iar fiecare revista este emisă într-o anumită lună. În stocurile bibliotecii există și culegeri de probleme rezolvate pentru care se cunoaște numele materiei și numele profesorului la examenul căruia s-au dat respectivele probleme.

Să se construiască clasa template GestionareImprumuturi care sa conțină informații despre diversele tipuri de articole ce pot fi împrumutate. Clasa conține cota articolului (incrementată automat la adăugarea unui nou articol), și o structură de obiecte, alocată dinamic care să gestioneze articolele împrumutate din bibliotecă.

Structura de date: unordered_set sau unordered map <cota_articol, structura de obiecte ce contine informatii despre articol>

- Să se adauge toate campurile relevante pentru modelarea acestei probleme.

Cerința suplimentară:

Să se construiască o specializare aferentă culegerilor de probleme rezolvate care să conțină sa conțină numărul total numărul total de culegeri aflate în stoc (incrementat la returul unei culegeri) și să afișeze toate culegerile aflate în prezent în stocul bibliotecii pentru materia Geometrie.

<u>Secțiunea de mai jos este aplicabilă tuturor temelor:</u>

BONUS: 1p care se adaugă la nota finală pe laborator pentru cele mai bune 3 lucrări practice din fiecare semigrupă care vor evidenția suplimentar conceptele de design patterns(https://refactoring.guru/design-patterns and

https://sourcemaking.com/design_patterns) și clean code standards (https://users.ece.cmu.edu/~eno/coding/CppCodingStandard.html).

În cazul în care nu sunt 3 lucrări într-o semigrupă care să evidențieze conceptele de mai sus, sloturile libere se alocă celeilalte semigrupe.