Lista temelor, cu toate cerintele, pentru **a doua lucrare practica**.

Cerinte comune tuturor temelor:

- toate clase vor conține obligatoriu constructori de inițializare, parametrizati și de copiere, destructor, iar operatorii „=”, „>>”, „<<” să fie supraincarcati.
- ilustrarea conceptelor: upcasting, downcasting, functii virtuale (pure – unde se considera mai natural)
- utilizarea variabilelor și functiilor statice
- citirea informațiilor complete a n obiecte, memorarea și afisarea acestora

Cerinte specifice fiecarei teme:

Tema 1. liste de numere reale, reprezentate ca tablouri uni- si bidimensionale:

Se dau urmatoarele clase:
- Vector (int dim, float *a)
- Matrice (Vector *v)
- Matrice_oarecare (int lin) : Matrice
- Matrice_patratica (int dim) : Matrice – nr de linii (dim) trebuie sa coincida cu dimensiunea oricarui vector component.

Obs. Se considera ca vectorii care intra în componenta matricelor au același numar de elemente.
Clasele derivate trebuie sa contina cel puțin constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza), destructori si o metoda care sa verifice daca o matrice triunghiulara este diagonala. Pentru matricile patratice, functia de afisare sa conțină și determinantul acestora.

Tema 2. vectori de numere complexe

Se dau urmatoarele clase:
- Complex (float re,im)
- Vector (int dim, Complex *v).
- Stiva : Vector - cu operatiile de adaugare și stergere modificate corespunzator.
- Coada: Vector - cu operatiile de adaugare și stergere modificate corespunzator.

Clasele derivate trebuie sa contina cel puțin constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza), destructor si o metoda prin care sa verifice dacă stiva / coada conține elemente care au doar partea imaginara nenula.

Tema 3. Cozi de caractere (implementate dinamic):

Se dau urmatoarele clase:
- Nod (char info, nod*next)
- Coada (nod * prim, nod * ultim, int dim_max);
- DEQUE : Coada - cu operatiile de adaugare și stergere modificate corespunzator.
- CoadaPrioritati (int *prioritate) : Coada - cu operatiile de adaugare și stergere modificate corespunzator.

Clasele derivate trebuie sa contina cel puțin constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza) si destructor. Simulați operatiile pe o stiva, respectiv pe o coada, folosind coada cu prioritati.

Tema 4. liste de numere întregi 1 (implementate dinamic)

Se dau urmatoarele clase:
- Nod (int info, nod*next)
- Nod_dublu (nod * ante) :Nod
- LDI (elemente de tip Nod_dublu); // lista dublu inlantuita;
- LSI : LDI ; // lista simplu inlantuita, obtinuta mostenind caracteriticile unei LDI adaptate la o inlantuire simpla;

Clasele derivate trebuie sa contina cel puțin constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor către constructorul din clasa de baza) si destructor. Să se exemplifice sortarea prin insertie directa utilizand LDI.

Tema 5. liste de numere întregi 2 (implementate dinamic)

Se dau urmatoarele clase:
- Nod (int info, nod*next)
- Nod_dublu (nod * ante) :Nod
- Lista (elemente de tip Nod_Prioritate)
- Lista_Circulara : Lista;

Clasele derivate trebuie sa contina cel puțin constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor către constructorul din clasa de baza) si destructor. Să se rezolve problema lui Josephus folosind liste circulare.

Tema 6. liste de numere întregi 3 (implementate dinamic)

Se dau urmatoarele clase:
- Nod (int info);
- Nod_simplu (nod * next) :Nod
- Nod_dublu (nod * ante) :Nod_simplu
- Nod_prioritate (int prio) :Nod_dublu
- Coada_prioritati (elemente de tip Nod_Prioritate);
- Coada_simpla : Coada prioritati (elementele au prioritati egale).

Clasele derivate trebuie sa contina cel puțin constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor către constructorul din clasa de baza) si destructorl. Să se exemplifice Heapsort utilizand coada cu prioritati și Insertionsort utilizand coada simpla.

Tema 7. multimi finite de numere întregi reprezentate ca tablouri unidimensionale

Se dau urmatoarele clase:
- Pereche(int prim,doi)
- Multime_pereche (int n, Pereche * p)
- Stiva_pereche : Multime_pereche – cu operatiile de adaugare și stergere modificate corespunzător;
- Coada_pereche : Multime_pereche – cu operatiile de adaugare și stergere modificate corespunzător;

Clasele derivate trebuie sa contina cel puțin constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza) si destructori. Să se implementeze o functie prin care se simuleaza operatiile pe o stiva de perechi folosind 2 cozi de perechi.

Tema 8. 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 9. 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

Clasele derivate trebuie sa contina cel puțin constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza), destructori si o metoda care sa verifice daca o matrice triunghiulara este diagonala. Pentru matricile patratice, functia de afisare sa conțină și determinantul acestora.

Tema 10. matrice de numere complexe reprezentate ca structuri inlantuite (ca matrici rare si nu neaparat patratice)

- Aceleași cerinte ca la Tema 9, fără calcularea determinantului.

Tema 11. arbori binari de cautare in reprezentare inlantuita:

Se dau urmatoarele clase:
- Nod (int info, nod*st, nod *dr)
- Nod_rosu_negru (char* culoare) :Nod
- Arbore(int nr_noduri)
- ABC (Nod *rad):Arbore
- Arbore_bicolor (Nod_rosu_negru *rad) : Arbore

Clasele derivate trebuie sa contina constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza), destructori si o functie virtuala prin care inaltimea arborelui bicolor sa fie identificata prin Adancimea neagra a radacinii.
Webografie: https://writer.zoho.com/public/miha_cio/Arbori-bicolori1/noband

Tema 12. arbori binari de cautare AVL in reprezentare inlantuita:

Se dau urmatoarele clase:
- Nod (int info, nod*st, nod *dr)
- Nod_AVL (int echilibru) :Nod
- Arbore(int nr_noduri)
- ABC (Nod *rad):Arbore
- Arbore_AVL(Nod_AVL *rad) : Arbore

Clasele derivate trebuie sa contina constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza), destructori si o functie virtuala prin care să se afiseze factorii de echilibru pentru fiecare nod.

Tema 13. arbori oarecare, in reprezentare inlantuita, prin legaturile fiu (catre fiul cel mai din stanga) si frate (catre urmatorul fiu al tatalui, in ordinea fiilor tatalui de la stanga la dreapta).

Se dau urmatoarele clase:
- Nod (int info)
- Nod_ABC: Nod(Nod*st, Nod*dr)
- Nod_fiu_frate (int copii, Nod ** leg) :Nod
- Arbore(int nr_noduri)
- ABC (Nod *rad):Arbore
- AB_oarecare(Nod_fiu_frate *rad) : Arbore

Clasele derivate trebuie sa contina constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza), destructori si o functie virtuala prin care afisarea elementelor unui arbore binar oarecare sa fie data de parcurgerea în latime, cu mentiunea listei fiilor pentru fiecare nod, iar afisarea unui ABC sa conțină cele 3 parcurgeri simultan.

Tema 14. grafuri

Se dau urmatoarele clase:
- Matrice (int **a) – matrice de adiacenta
- Vector (int *v, int dim)
- Lista (Vector *l) – lista de adiacenta
- Graf (int nr_noduri) – clasa abstracta
- Graf_Neorientat(Lista L) : Graf
- Graf_Orientat (Matrice A) :Graf

Clasele derivate trebuie sa contina constructor parametrizat (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza), destructor si o metoda care sa afiseze vectorul de tati, dacă pentru un Graf Orientat se verifica conexitatea lui și se da un nod de plecare pe post de rădăcina.

Tema 15. grafuri orientate,

Se dau urmatoarele clase:
- Matrice (int **a) – matrice de adiacenta
- Graf (int nr_noduri) – clasa abstracta
- Graf_complet (int nr_muchii) : Graf
- Graf_antisimetric(Matrice M) : Graf
- Graf_turneu: Graf_complet, Graf_antisimetric

Clasele derivate trebuie sa contina constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa/clasele de baza), destructori si o metoda virtuala care sa afiseze arcele unui Graf_antisiemtric, sau a unui Graf Turneu.

Obs: graf turneu – orientat complet si antisimetric. Un graf orientat se numeşte antisimetric dacă pentru oricare două vârfuri din graf x şi y dacă există arcul (x,y), atunci nu există arcul (y,x). Un graf orientat sau neorientat se numeşte complet dacă oricare două vârfuri din graf sunt adiacente.

Tema 16: Din clasa Nr_Natural_Mare să se deriveze clasa

Se dau urmatoarele clase:
- Vector (int *a)
- Nr_Natural_Mare(int_nrcif, Vector V)
- Numar_Intreg_mare(char semn) : Nr_Natural_Mare

Să se implementeze operatiile uzuale. Clasele derivate trebuie sa contina constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza) si destructori.

Tema 17.

Se dau urmatoarele clase:
- Vector (int *a)
- Nr_Natural_Mare(int_nrcif, Vector V)
- Numar_Fibonacci_mare:Nr_Natural_Mare

Sa se implementeze ierarhia de clase mai sus mentionata, iar clasa derivata trebuie sa contina constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa baza), destructor si o functie virtuala care sa afiseze numarul Fibonacci, precum si descompunerea lui in suma de numere Fibonacci.

Tema 18.

Departamentul „Salarizare” al firmei X actualizeaza fisele intregului personal utilizand o aplicație OOP. In acest scop se vor implementa clasele:

- Data (int zi, char* luna, int an);
- Angajat (char *nume, char *prenume , float salariu, Data data_angajarii);
- Part_Time (int nr_ore_zi, Data final_contract) : Angajat
- Permanent (int nr_minori_intretinere) : Angajat

De sărbători fiecare angajat va primi o prima. Angajatul permanent va primi un spor calculat ca un procent din prima fixă, egal cu vechimea lui în firmă, calculată până la 31 decembrie 2018 (de ex., dacă vechimea în firmă este de 12 ani, atunci va primi pentru fiecare copil minor 12% * suma standard stabilita de firma etc), iar angajații part time vor primi suma standard

stabilită, dacă contractul lor nu se termina la sfârșitul lunii curente, caz în care primesc doar 75% din suma standard prevazuta.

Realizați programul OOP astfel încât departamentul de salarizare sa aibă informații complete despre toți angajații.

Clasele derivate trebuie sa contina constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza) si destructori.

Tema 19.

Se dau urmatoarele clase:
- Clasa Persoana(int id, char* nume)
- Clasa Abonat:Persoana(char *nr_telefon)
- Clasa Abonat_Skype: Abonat (id_skype)
- Clasa Abonat_Skype_Romania (adresa_mail) : Abonat_Skype
- Clasa Abonat_Skype_Extern(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 .

Clasele derivate trebuie sa contina constructori parametrizati (prin care sa se evidentieze transmiterea parametrilor catre constructorul din clasa de baza) si destructori.

Tema 20

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 (flot 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.