# Test PC2 - subiect 3 – dec. 2023

Să se implementeze următoarele:

(0.5p) O clasă *Produs* ce are un singur membru:
- Un membru **producător** de tip **string**

(0.5p) O clasă *Electrocasnic* **derivată** din clasa *Produs* ce conține:
- Un membru **clasaEnergetica** de tip **string**
- Un membru **greutate** de tip **unsigned int**

Pentru clasa *Produs* se vor implementa următoarele metode:
- (0.5p) Constructor de inițializare (cu listă de parametri) care va inițializa membrul clasei cu valoarea parametrului
  - Acest constructor va permite și apelarea sa fără parametri, caz în care membrii se vor inițializa cu valori predefinite
- (0.5p) Metoda de afișare a tuturor datelor referitoare la un produs.

Pentru clasa *Electrocasnic* se vor implementa următoarele metode:
- (0.5p) Constructor de inițializare (cu listă de parametri) care va inițializa membrii clasei cu valorile parametrilor și apelează un constructor din clasa de bază pentru inițializarea membrului mostenit
  - Acest constructor va permite și apelarea sa fără parametri, caz în care membrii se vor inițializa cu valori predefinite
- (0.5p) Metoda de afișare a tuturor datelor referitoare la un electrocasnic.

Cerințe:
- (0.4p) Creați 2 obiecte de tip *Produs* pentru a testa cele două moduri de apelare a constructorului din clasă.
  - (0.1p) Pentru aceste două obiecte apelați funcția de afișare
- (0.4p) Creați 2 obiecte de tip *Electrocasnic* pentru a testa cele două moduri de apelare a constructorului din clasă.
  - (0.1p) Pentru aceste două obiecte apelați funcția de afișare

Barem

| | |
|---|---|
| 1. Clasa Produs | |
| 1.1. Definirea corectă a datelor membre | 0.5 |
| 1.2. Constructor | 0.5 |
| 1.3. Metoda de afișare | 0.5 |
| 2. Clasa Electrocasnic | |
| 2.1. Implementarea corectă a conceptului de moștenire + date membre | 0.5 |
| 2.2. Constructor | 0.5 |
| 2.3. Metoda de afișare | 0.5 |
| Crearea celor 2 obiecte Produs + Afișare | 0.5 |
| Crearea celor 2 obiecte Electrocasnic + Afișare | 0.5 |
| Oficiu | 1 |
| TOTAL | 5 |

**Vezi și pagina 2!**

**NUMAI DUPĂ** implementarea cerințelor de pe pagina 1 vă puteți apuca de cerințele de mai jos:

1.
    a. (0.5p) Supraîncărcați un **operator binar** care schimbă greutatea unui obiect Electrocasnic.
    b. (0.5p) Testați și demonstrați funcționarea operatorul scris în funcția main.

2. Creați o nouă clasă *Aspirator* **derivată** din clasa *Electrocasnic* ce va avea:
    - Un membru **putere** de tip unsigned int
    - (0.5p) Constructor de inițializare
    - (0.5p) Metoda de afișare

3.
    a. (0.5p) Adăugați în funcția main un vector din biblioteca standard care să permită adăugarea a două obiecte de tip *Electrocasnic* și două obiecte de tip *Aspirator*. Se pot folosi obiectele existente sau se pot adăuga altele noi.
    b. (0.5p) Parcurgeți vectorul și afișați obiectele din acesta. Pentru fiecare obiect din vector trebuie să se afișeze toate elementele sale.

4. (1p) Dați un exemplu de utilizare a unui **operator=** supraîncărcat printr-o funcție proprie pentru una din cele 3 clase.

5. (1p) Dați un exemplu de utilizare a **pointerilor smart** pentru orice obiect sau tip de date definit anterior.

**Observații**:
- Se va utiliza specificatorul de acces "public" cel mult pentru metodele claselor (nerespectarea acestei prevederi atrage după sine înjumătățirea punctajului acordat).
- Fiecare clasa va avea headerul + sursele proprii, iar definirea metodelor se va realiza în fișierul sursă (.cpp)
- Clasele se pot completa, la nevoie, cu alte metode ajutătoare
- Numele membrilor claselor poate fi modificat dacă se dorește utilizarea unor convenții de notare