Moștenirea multiplă și agregarea

Sensei-ul ninja-ului nostru le-a ținut elevilor săi o predică despre importanța sticlelor care conțin apă. La final elevii au primit ca și task să mediteze într-un loc înconjurați de 5 mese, pe fiecare dintre ele fiind așezată o sticlă. Una din cele 5 sticle conține otravă, restul fiind umplute cu apă potabilă, iar task-ul este să o taie cu sabia pe cea cu otravă, dar nu oricum, ci legați la ochi folosindu-se de propriile instincte și simțuri.

Cum NullPointer este pasionat și de programare și-a propus în același timp să învețe și moștenirea multiplă cât și relația de incluziune („has-a"). Așadar înainte de a face exercițiul primit de la senseiul său trebuie să îl învățăm noi puțină programare. Prin urmare vom considera următoarele clase în limbajul C++:

```
class Recipient
{
    float volum;
    char* material;
};

class Produs
{
    double pret;
    char* producator;
};

class Sticla
{

};
```

```
class Dop
{
    float diametru;
    char* material;
};
```

Cerințe

1) Să se stabilească corect relațiile de is-a și respectiv has-a pentru clasele de mai sus. (1p)

2) Pentru fiecare clasă să se implementeze constructorii fără parametri și constructorii cu toți parametrii. (2p)

3) Să se implementeze constructorii de copiere și operatorii de asignare pentru fiecare clasă. (2p)

4) Să se implemeteze destructorii pentru clasele care necesită dezalocarea resurselor alocate dinamic. (1p)

5) Să se implementeze operatorul de afișare pentru fiecare clasă din cele de mai sus. (1p)

6) În funcția main să se creeze un vector alocat dinamic cu minimum 5 elemente de tip Sticla și pe urmă afișați-l. Elementele nu se vor citi de la tastatură ci se vor hardcoda direct în funcția main. (1p)

7) Sortați vectorul de Sticle (crescător sau descrescător) după diametrul dopurilor și apoi afișați-l. Pentru rezolvarea acestei cerințe se va supraîncărca după caz unul din operatorii < sau >. Pentru acest vector se vor crea 3 funcții definite într-un namespace denumit SortingHelper. (1p)

Se acordă un punct din oficiu. Succes.

Precizări

- Clasa se va implementa în fișiere .h și .cpp. Iar testarea codului se va face în funcția main care va fi într-un fișier separat.

- Neimplementarea în fișiere .h și .cpp atrage după sine pierderea unui punct.

- Nu este permisă marcarea câmpurilor claselor părinte cu protected.

- Erorile de compilare și memory leak-urile atrag fiecare după sine pierderea unui punct.

- Alte depunctări cu privire la modul de implementare sau la corectitudinea declarării metodelor, rămân la latitudinea asistentului care vă corectează.

- Dacă nu s-a specificat vreun mod de implementare a cerințelor rămâne la latitudinea voastră cum vreți să procedați în rezolvare.