Primer examen LP 2021-2022

Laboratori de programació (102767)

Exàmen LP 1er parcial 2021-22

Nom..... Cognoms.....

NIU..... Grup .....

Grups: 41(Javier); 43(Gemma); 45-1;51-1(Marcel); 45-2;51-2(Roberto); 45-3(Estrella)

Ex1	Ex2	Ex3	FINAL

EXERCICI 1. (3.5 pts) TEMPLATES.

Tenim una classe Magatzem que ens permet desar productes a un magatzem fins a arribar a un volum màxim. Inicialment aquest magatzem el teníem definit per un tipus de productes anomenat A. Un producte de tipus A tenia un atribut que era un codi que l'identificava i era únic i més atributs entre ells el volum. Ara volem que aquesta **classe Magatzem** serveixi per qualsevol tipus de producte -no només pels A-, i en concret la voldrem utilitzar per Mobles. Aquí tenim una primera versió de la classe Magatzem per desar elements de tipus A.

Suposem que tenim la **classe Magatzem** definida de la següent manera:

```
1.
2. class Magatzem
3. {
4.  public:
5.     Magatzem(float capacitat=10000) :m_capacitat(capacitat) {};
6.     ~Magatzem();
7.     void afegirProducte(A& m);
8.     void eliminarProducte(A& m);
9.     friend ostream& operator<< (ostream& out, const Magatzem& m)
10.    {      out << "Magatzem: " << endl;
11.           for (auto it = m.m_contingut.begin(); it != m.m_contingut.end(); it++)
12.           {   out << "Producte: " << (*(it->second)) << endl;          }
13.           return out;
14.    }
15.  private:
16.    unordered_map<int,A*> m_contingut;
17.    float m_capacitat;
18.};
19.
20. void Magatzem::afegirProducte(A& m)
21  {  if (m_capacitat + m.Volum() >= 0)
22.    {      m_contingut[m.Codi()] = new A(m); }
23. }
24.
25. void Magatzem::eliminarProducte(A& m)
26. {
27.    for (auto it = m_contingut.begin(); it != m_contingut.end(); it++)
28.    {      if (it->first == m.Codi())
29.           {      delete (it->second);
30.                  m_contingut.erase(it);
31.           }
32.    }
33. }
34.
35. Magatzem::~Magatzem()
36. {
37.    for (auto it = m_contingut.begin(); it != m_contingut.end();it++ )
38.    {      delete (it->second);              }
39. }
```

(NOTA: Suposeu que TOTS els includes necessaris ja estan fets)

1. **(1.8 punts)** Modifiqueu la classe Magatzem per a que sigui un *template.* Podeu posar només les línies que heu modificat.

2. **(1.1 punts)** Donada la classe Moble definida com es veu a continuació. Definiu i implementeu només els mètodes que calgui a Moble per tal de poder crear un Magatzem de Moble i poder utilitzar tots els seus mètodes.

```
class Moble
{
public:
    Moble();
    ~Moble() {}

private:
    int m_codi;
    float m_volum;
};
```

3. **(0.6 punts)** Tenim definit un Moble m1 de la següent manera. Defineix un Magatzem utilitzant el template Magatzem definit, que es digui **magatzemLP**, afegeix un moble amb codi 1, i imprimeix el magatzem per pantalla.

```
Moble mob(1,456.7);
```

EXERCICI 2. (5 pts) HERÈNCIA.

Ens han encarregat fer l'aplicació de gestió d'una empresa de lloguer de vehicles. Com que ja som programadors experimentats, decidim fer l'aplicació orientada a objectes i utilitzant herència.

Els vehicles que es poden llogar han de ser d'un dels següents tipus:

- Turisme
- Furgoneta

Per qualsevol vehicle, sigui del tipus que sigui, hem de tenir la següent informació: **matrícula**, **marca**, si està **disponible** (sí/no) i el **preu de lloguer** per dia.

A més, els següent tipus de vehicle tenen informació addicional:

- **Turisme**: total de places
- **Furgoneta**: tipus (industrial (I) o de viatgers (V))

El preu de lloguer diari té un taxa fixa que, per defecte és de **10 €,** però que es pot modificar amb el mètode **setTaxaFixa()**. Aquesta taxa fixa s'aplica a tots els vehicles sigui quin sigui el vehicle. A més, depenent del vehicle hi ha un cost afegit diari segons la següent taula:

Tipus de vehicle	Cost afegit de lloguer diari
Turisme	5 € per cada plaça
Furgoneta	Industrial: 60 € / Viatgers: 40€

La declaració de la classe **Vehicle** és com es mostra a continuació:

```
class Vehicle {
public:
	Vehicle(const string& matricula, const string& marca,bool disponible) :
		m_matricula(matricula), m_marca(marca), m_disponible(disponible),
		m_taxaFixa(10) {};
	~Vehicle();
	void setMatricula(const string& matricula) { m_matricula = matricula; };
	string  getMatricula() const { return m_matricula; };
	void setMarca(const string& marca) { m_marca = marca; };
	string  getMarca() const { return m_marca; };
	void setDisponible(bool disponible) { m_disponible = disponible; };
	bool  getDisponible() const { return m_disponible; };
	void setTaxaFixa(float taxaFixa) { m_taxaFixa = taxaFixa;};
	float getPreu() const;
private:
	string m_matricula;
	string m_marca;
	bool m_disponible;
	 float m_taxaFixa;
};
```

1- (1 punt) Sabent que tant **Turisme** com **Furgoneta** són també vehicles i comparteixen tots els atributs de la classe **Vehicle**, completa a continuació la implementació del codi de les classes **Turisme i Furgoneta** on aparegui de cada classe **només**:

- el necessari perquè quedi reflectit si alguna classe hereta d'alguna altra.
- el constructor
- els membres privats de la classe

2.- (1.5 punt) Declara i implementa els mètodes **getPreu()** de totes les classes. Fes les adaptacions necessàries perquè una crida al mètode getPreu() de qualsevol classe, torni el preu de lloguer diari que correspongui segons el tipus i les característiques del vehicle:

3.- (1 punt) Crea una llista per desar els diferents vehicles. A continuació, recorre la llista de cotxes i mostra, dels vehicles **disponibles**, la matrícula, la marca i el preu de lloguer de cadascú. Els vehicles que volem tenir a la llista seran els següents (però el codi ha de servir per qualsevol llista de les mateixes característiques).

Vehicle: Turisme; Matricula: 5678-GLK; Marca: Peugeot; Preu Lloguer: 35; Disponible: True

Vehicle: Turisme; Matricula: 2137-LHB; Marca: Smart; Preu Lloguer: 20; Disponible: True

Vehicle: Turisme; Matricula: 8126-KFD; Marca: Opel; Preu Lloguer: 70; Disponible: True

4.- (1.5 punt) Per seguretat, volem crear en memòria una còpia de la llista de vehicles creada a l'apartat anterior. Afegeix els mètodes necessaris a la declaració de les classes Vehicle, Turisme i Furgoneta i fes la seva implementació. Un cop fet això crea la nova llista i copia els elements de la primera llista a la segona.

Nota: No cal implementar els constructors de còpia, els que ens crea el compilador per defecte ja ens van bé.

EXERCICI 3. (1.5 pts) HASH EXERCICI . (1.5 pts) HASH.

Tenim una taula Hash amb 9 posicions parcialment plena (**TAULA**). La clau és un enter i el valor una paraula. Volem fer les següents insercions i esborrats. La funció Hash és **h(clau)=clau%9**. La manera de resoldre col·lisions és un sondeig quadràtic **h(clau)=(h(clau)+i²)%9** a on **i** és el nombre d'intents de reposicionar la parella <clau, valor>.

1. **(0.6 pts)** Per cada operació indiqueu per quines caselles successives passeu (mireu càlculs fets dels mòduls), perquè les rebutgeu: L:Lliure, O:ocupada, E:Esborrada. Fins a quina posició arribeu i perquè us quedeu a ella: L,O,E. Si hi ha alguna raó extra per quedar-se expliqueu-la. Per la cerca indiqueu també a la cinquena columna a omplir si s'ha trobat l'element o no. Finalment indiqueu com queda la taula al final de totes les operacions modificant la **TAULA** que us donem(essent L:Lliure, O:Ocupat,E:Esborrat). Si heu d'esborrar un element indiqueu-lo a la taula final com a tatxat.
   Recordeu que el mòdul és el residu de la divisió entera: 20%9=2; 4%9=4 etc...

Alguns càlculs fets..... (12%9)=3; (31%9)= 4; (5%9)=6; (4+ 2^2)%9=8

Operació	**<clau,valor>**	**Càlcul Posició Inicial**	**Posicions successives i perquè les rebutges: E: està esborrada, L:Està lliure; O:està ocupada.**	**Posició final i pq la tries: L,O,E**
Inserir	**<20,"Carme">**			
Cercar	**<4, "Miquel">**			
Esborrar	**<12,Cristina>**			
Esborrar	**<31,"Albert">**			

TAULA INICIAL:

Posició	**< Clau , Valor>**	**Estat**
0	<23,Julia>	**O**
1	<19,Martí>	**O**
2		**L**
3	<3,Marta>	**O**
4	<12,Cristina>	**O**
5	<14,Gemma>	**O**
6	<6,Guillem>	**O**
7	<25,Xavier>	**O**
8		**L**