Grupo I

a) Diga quais são os métodos construtores que estarão presentes em cada uma das classes... [1.4 val.]

Na classe I:
- Construtor por defeito
- Construtor de cópia

Na classe A:
- Construtor por defeito
- Construtor de cópia

Na classe B:
- Construtor por defeito
- Construtor de cópia
- Construtor de conversão `B(A &)`

b) Apresente o resultado que será visualizado na saída standard com a execução do programa. [2.7]

```
<1>
novo A 1
novo A 2
novo B
<2>
<3>
B::print 0
A::print 1
A::print 2
<4>
novo A 3
<5>
novo A 4
novo A 5
A => B 3
<6>
B 3 eliminado
A 5 eliminado
A 4 eliminado
A 3 eliminado
B 0 eliminado
A 2 eliminado
A 1 eliminado
```

c) O que seria visualizado se o método *`get`* da classe *`I`* não fosse virtual? Refira-se apenas... [0.5 val.]

Seria visualizado exatamente o mesmo.

d) O que seria visualizado se o método *`print`* da classe *`I`* não fosse virtual? Refira-se apenas... [0.5 val.]

```
...
<3>
I::print
<4>
...
```

e) E o que seria visualizado se o método destrutor da classe *`I`* não fosse virtual? Refira-se... [0.5 val.]

Seria visualizado exatamente o mesmo que em b).

f) Diga se as declarações *I V1[10]; I *V2[20];* são ou não válidas no problema considerado... [1.2 val.]
São ambas válidas;
São invocados 10 construtores (na instanciação dos 10 objetos do vetor V1).

g) Faça as alterações que julgue necessárias para que a função *main* ... if (a>0) cout<<a; [1.2 val.]

```
bool A::operator>(int v)const{return val>v;}

ostream &operator<<(ostream &o, const A &a){
	o<<a.get();
	return o;
}
```

GRUPO II

a)

```
class Veiculo{
	string matricula;
	Colecao<Passagem *> passagens;
public:
	Veiculo(string m): matricula(m){}

	string getMatricula()const {return matricula;}

	bool addPassagem(Passagem *p){return passagens.insert(p);}

	bool operator<(const Veiculo &outra)const {return matricula<outra.matricula;}
};


class VLigeiro: public Veiculo{
public:
	VLigeiro(string m): Veiculo(m){}
};

class VPesado: public Veiculo{
	int tara;
public:
	VPesado(string m, int t): Veiculo(m){tara=t;}
};


class Passagem{
	string datahora;
	Veiculo* veiculo;
	Portico* portico;
public:
	Passagem(string dh, Veiculo* v, Portico* p): datahora(dh){
		veiculo=v;
		portico=p;
	}

	bool operator<(const Passagem &outra)const {
		if (datahora!=outra.datahora) return datahora<outra.datahora;
		else return veiculo->getMatricula()<outra.veiculo->getMatricula();
	}
};
```

```
class Portico{
	string local;
	Colecao<Passagem> passagens;
public:
	Portico(string loc){local=loc;}
	bool addPassagem(string dh, Veiculo *v){
		Passagem p(dh,v,this);
		if (passagens.insert(p)) return v->addPassagem(passagens.find(p));
		else return false;
	}

	bool operator<(const Portico &outro)const {return local<outro.local;}
};


class GestPort{
	ColecaoHibrida<Veiculo*> veiculos;
	Colecao<Portico> porticos;
public:
	bool addPortico(string loc){
		Portico p(loc);
		return porticos.insert(p);
	}

	bool addVLigeiro(string mat){
		Veiculo *v = new VLigeiro(mat);
		return veiculos.insert(v);
	}

	bool addVPesado(string mat, int t){
		Veiculo *v = new VPesado(mat,t);
		return veiculos.insert(v);
	}

	Portico *findPortico(string loc){
		Portico p(loc);
		return porticos.find(p);
	}

	Veiculo *findVeiculo(string mat){
		Veiculo v(mat);
		return veiculos.find(&v);
	}

	bool addPassagem(string dh, string mat, string loc){
		Veiculo *v=findVeiculo(mat);
		if(v!=NULL){
			Portico *p=findPortico(loc);
			if(p!=NULL) return p->addPassagem(dh,v);
			else return false;
		}else return false;
	}
};
```

b)

```
void main(){
	GestPort a;
	a.addPortico("A24 Km 53");
	a.addVLigeiro("11-22-AA");
	a.addVPesado("33-44-BB", 21000);
	a.addPassagem("1/2/12 19:00", "33-44-BB", "A24 Km 53");
}
```

GRUPO III

a)

```
	void GestPort::printVeiculos()const{
		ColecaoHibrida<Veiculo *>::iterator it;
		for(it=veiculos.begin(); it!=veiculos.end(); it++)
			(*it)->print();
	}

	virtual void Veiculo::print()const=0;

	void VLigeiro::print()const{ cout<<"Veiculo Ligeiro "<< getMatricula();}

	void VPesado::print()const{
		cout<<"Veiculo Pesado "<< getMatricula()<<" com tara "<< tara << endl;
	}
```

b)

```
	void GestPort::printPassagensPortico(string loc){
		Portico *p=findPortico(loc);
		if(p!=NULL) p->printPassagens();
	}

	void Portico::printPassagens()const{
		cout<<"Portico "<< local<<":\n";
		Colecao<Passagem>::iterator it;
		for(it=passagens.begin(); it!=passagens.end(); it++)
			it->print();
	}

	void Passagem::print()const{
		cout<<"Veiculo "<< veiculo->getMatricula()<< " passou a "<<datahora<<endl;
	}
```

c)

```
	class Portico{
		float tarifa;
		...
	public:
		...
		void setTarifa(float val){tarifa=val;}
	};

	bool GestPort::novaTariva(float val, string loc){
		Portico *p=findPortico(loc);
		if(p!=NULL) {p->setTarifa(val); return true;}
		else return false;
	}
```

d)

```
float GestPort::totalFaturado()const{
	float tot=0;
	Colecao<Portico>::iterator it;
	for(it=porticos.begin(); it!=porticos.end(); it++)
		tot+=it->totalFaturado();
	return tot;
}

float Portico::totalFaturado()const{
	float tot=0;
	Colecao<Passagem>::iterator it;
	for(it=passagens.begin(); it!=passagens.end(); it++)
		tot+=it->numTarifas()*tarifa;
	return tot;
}

int Passagem::numTarifas()const{ return veiculo->numTarifas();}

virtual int Veiculo::numTarifas()const=0;

int VLigeiro::numTarifas()const{return 1;}

int VPesado::numTarifas()const{ return 2 + tara/5000;}
```

e)

```
GestPort::~GestPort (){
	ColecaoHibrida<Veiculo *>::iterator it;
	for(it=veiculos.begin(); it!=veiculos.end(); it++) delete *it;
	veiculos.clear();
}
```

f)

```
bool GestPort::remVeiculo(string mat){
	Veiculo *v=findVeiculo(mat);
	if(v!=NULL && v->numPassagens()==0){
		veiculos.erase(v);
		delete v;
		return true;
	}else {cout<<"Remocao mal sucedida\n"; return false;}
}


int Veiculo::numPassagens()const {return passagens.size();}
```

g)

```
void main(){
	GestPort a;
	a.addPortico("A24 Km 53");
	a.addVLigeiro("11-22-AA");
	a.addVPesado("33-44-BB", 21000);
	a.addPassagem("1/2/12 19:00", "33-44-BB", "A24 Km 53");
	a.printVeiculos();
	a.printPassagensPortico("A24 Km 53");
	a.novaTariva(3,"A24 Km 53");
	cout<<"Total faturado: "<<a.totalFaturado()<<endl;
	a.remVeiculo("11-22-AA");
}
```