Engenharia Informática/Informática de Gestão
Época Normal – 30 de janeiro de 2019

Grupo I (14 valores)

a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos ... [11.4 val.]

```
class Email{
	int id;
	string data;
	string texto;
	Conta* remetente;
	Conta* destinatario;
public:
	Email(int id, string txt, Conta *remet, Conta *dest): texto(txt){
		this->id = id;  remetente = remet; destinatario = dest;
		data = agora();
	}

	bool operator<(const Email &outro)const {
		return id < outro.id;
	}
};


class Conta{
	string eemail;
	string utilizador;
	Colecao<Email> inbox;
	Colecao<Email*> enviadas;
public:
	Conta(string ee, string ut) : eemail(ee), utilizador(ut) {}
	bool addEmailEnviado(Email *em) {
		return enviadas.insert(em);
	}
	bool addEmailRecebido(const Email &em) {
		return inbox.insert(em);
	}
	Email *findEmailRecebido(const Email &em){
		return inbox.find(em);
	}
	int numEmailsRecebidos()const { return inbox.size(); }
	bool operator<(const Conta &outra)const {return eemail<outra.eemail;}
};


class GestEmail{
	Colecao<Conta> contas;
public:
	bool registarConta(string ee, string ut){
		Conta c(ee, ut);
		return contas.insert(c);
	}
```

```
	bool enviarEmail(string eeremet, string eedest, string texto) {
		Conta *rem = findConta(eeremet);
		if (rem != NULL) {
			Conta *dest = findConta(eedest);
			if (dest != NULL) {
				int id = dest->numEmailsRecebidos()+1;
				Email em(id, texto,rem,dest);
				if (dest->addEmailRecebido(em))
				return rem->addEmailEnviado(dest->findEmailRecebido(em));
				else return false;
			} else return false;
		} else return false;
	}
private:
	Conta *findConta(string ee){
		Conta  c(ee, "");
		return contas.find(c);
	}
};
```

b) Acrescente ao problema os métodos que permitam apresentar na saída .... [1.8 val.]

```
void GestEmail::mostrarEmailsRecebidos(string ee) {
	Conta *c = findConta(ee);
	if (c != NULL) c->mostrarEmailsRecebidos();
}
void Conta::mostrarEmailsRecebidos() {
	Colecao<Email>::iterator it;
	for (it = inbox.begin(); it != inbox.end(); it++)
		it->print();
}
void Email::print() const {
	cout << "De: " << remetente->getEEmail() << "    Data: " << data;
	cout << texto << endl;
}
string Conta::getEEmail() const { return eemail; }
```

c) Implemente um pequeno *main* que faça uso de todas as funcionalidades ... [0.8 val.]

```
void main(){
	GestEmail ge;
	ge.registarConta("arita@ipb.pt", "Ana Rita");
	ge.registarConta("asofia@ipb.pt", "Ana Sofia");
	ge.enviarEmail("arita@ipb.pt", "asofia@ipb.pt", "Ola!");
	ge.mostrarEmailsRecebidos("asofia@ipb.pt");
}
```

Grupo II (6 valores)

a) Quais das classes dispõem do construtor por defeito? E quais delas dispõem do construtor de cópia? [0.8 val.]

Apenas a classe Animal dispõe do construtor por defeito, e em todas as classes está presente o construtor por cópia.

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

```
Novo Animal de nome Boby
Novo Domestico
-1-
Novo Animal de nome Lacy
-2-
Novo Animal de nome Kitty
Novo Domestico
-3-
->Bicho mimado->Boby
-4-
->Lacy
-5-
->Bicho mimado->Kitty
-6-
->Kaputt->Xau Boby->Menos um animal
-7-
->Xau Lacy->Menos um animal
-8-
->Kaputt->Xau Kitty->Menos um animal
```

c) Apresente o resultado que seria visualizado na saída standard se o método `print()` ... [0.8 val.]

```
-3-
->Indiferenciado
-4-
```

d) Tendo em conta as classes definidas no problema, dê uma breve explicação do erro cometido em cada uma das instruções que se seguem: [0.9 val.]

(1) `Animal *a = new Animal("Farrusco");`

A classe Animal não dispõe do construtor de conversão que está a ser usado na sua instanciação.

(2) `Animal *a = new Domestico();  Domestico *d; d = a;`

A conversão descendente (downcast) que se está a tentar fazer na atribuição `d = a` só seria possível na forma explícita, usando um operador de conversão.

(3) `Domestico d("Farrusco"); DeEstimacao &de = d;`

Também aqui se está a tentar fazer uma conversão descendente, só possível, como dissemos, na forma explícita.

e) Faça as alterações que achar necessárias nas classes do problema para que a linha de código que se segue possa surgir na função *main*. [0.8 val.]

```
DeEstimacao::DeEstimacao(): Domestico("") {}
```

Outra opção válida, seria:

```
DeEstimacao::DeEstimacao(){}
Domestico::Domestico(): nome("") {}
```