CI1 Parte prática. Duração: 2h00m

Pretende-se implementar um sistema de gestão da correspondência numa estação de correio. A correspondência (classe **Mail**) é identificada por nome do remetente (*sender*), nome do destinatário (*receiver*) e código postal do destinatário (*zipCode*). Por uma questão de simplificação, a correspondência considerada neste cenário tem peso máximo de 2Kg, só é efetuado o envio para território nacional e existem apenas dois tipos de envio: correio normal e correio verde.

O correio normal é caracterizado por mais um atributo, que é o peso (*weight*) da correspondência e o correio verde pelo tipo de embalagem usada (*envelope*, *bag*, *box*). As classes **RegularMail** e **GreenMail** caracterizam os dois tipos de correspondência enumerados.

A classe ***Postman*** identifica um carteiro, e inclui um identificador (*id*), o nome do carteiro (*name*) e a correspondência que este está encarregue de distribuir (*myMail*).

A classe ***PostOffice*** identifica uma estação de correio e inclui informação sobre os carteiros que aí trabalham (*postmen*), gama de códigos postais da localidade que a estação serve ([*firstZipCode*, *lastZipCode*]), correspondência entregue pelos clientes para envio para outras estações de correio (*mailToSend*) e correspondência a ser distribuída na localidade (*mailToDeliver*).

As classes **PostOffice, Postman, Mail, RegularMail** e **GreenMail** estão parcialmente definidas a seguir.

NÃO PODE acrescentar membros-dado nas classes **Mail**, **RegularMail** e **GreenMail.**

```
class Mail {
   string sender;
   string receiver;
   string zipCode;
public:
   Mail(string send, string rec, string
                              zcode);
   // …
};

class RegularMail: public Mail {
    unsigned int weight;
public:
    RegularMail(string send, string rec,
           string code, unsigned int w);
};

class GreenMail: public Mail {
    string type;  // "envelope", "bag", "box"
public:
    GreenMail(string send, string rec,
            string code, string t);
};
```

```
class Postman {
   unsigned int id;
   string name;
   vector<Mail *> myMail;
public:
   Postman();
   // …
};


class PostOffice {
   vector<Mail *> mailToSend;
   vector<Mail *> mailToDeliver;
   vector<Postman> postmen;
   string firstZipCode, lastZipCode;
public:
   PostOffice();
   PostOffice(string fZCode, string lZCode);
   // …
};
```

a) [**2.5 valores**] Implemente a função template seguinte:

unsigned int numberDifferent (const vector<T> & v1)

Esta função retorna o número de elementos diferentes existentes no vetor `v1`, passado por argumento. Considere que `v1` é um vetor não ordenado.

b) [**3 valores**] Implemente na classe **PostOffice** o membro-função:

*vector<Mail *> removePostman(string name)*

Esta função remove do vetor `postmen` o carteiro de nome `name`, retornando a correspondência que estava a cargo desse carteiro (`myMail`). Se não existe nenhum carteiro de nome `name`, a função retorna um vetor vazio.

c) [**2.5 valores**] Implemente, nas classes que considerar necessário, o membro-função que efetua o cálculo do preço do selo, em cêntimos, para envio de uma correspondência:

unsigned int getPrice () const

Esta função retorna o preço do envio da correspondência, sabendo que:

- o custo de envio de correspondência por correio normal depende do seu peso (p):

  50 cent, se p<=20g ; 75 cent, se 20<p<=100g ; 140 cent, se 100<p<=500g ; 325 cent, se p>500
- o custo do envio de correspondência por correio verde depende do tipo de embalagem usada:

  80 cent, se envelope (*envelope*) ; 200 cent, se saqueta (*bag*) ; 240 cent, se caixa de cartão (*box*)

d) [**3.5 valores**] Implemente na classe **PostOffice** o membro-função:

*vector<Mail *> endOfDay(unsigned int &balance)*

No final do dia, o responsável pela estação de correios deve:

- verificar se o valor em caixa está correto. Esta função calcula o preço total relativo a toda a correspondência entregue na estação de correios (vetor *mailToSend*) e coloca esse valor no argumento *balance*

- tratar a correspondência entregue na estação de correios (esvaziar o vetor *mailToSend*), sendo: a correspondência a entregar pelos carteiros adicionada ao vetor *mailToDeliver* (se *zipCode* está na gama dos códigos postais da estação ([*firstZipCode*, *lastZipCode*]) ; a correspondência a enviar para outras estações colocada num vetor que é o retorno da função *endOfDay*. Nota: A classe *PostOffice* já possui o membro-função `void addMailToDeliver(Mail *m)`

e) [**2.5 valores**] Implemente na classe **Postman** o construtor:

Postman(string name)

Este construtor recebe como argumento o nome (*name*) do carteiro. Atribui a cada objeto carteiro criado um novo identificador (*id*), sequencial e incremental. O primeiro carteiro a ser criado terá *id* igual a 1, o segundo *id* igual a 2 e assim sucessivamente. O vetor da correspondência a ser distribuída pelo carteiro (*myMail*) está vazio.

f) [**3 valores**] Implemente na classe **Postman** o operador <. Um carteiro `p1` é menor que outro carteiro `p2`, se o número de locais distintos a que tem de se deslocar para distribuir a sua correspondência (*myMail*) é inferior. A identificação de um local é dada pelo código postal (`zipCode`) da correspondência. Considere que o vector `myMail` se encontra ordenado por código postal da correspondência.

Sugestão: se realizou a alínea a), utilize a função template aí implementada.

g) [**3 valores**] Implemente na classe **PostOffice** o membro-função:

*Postman addMailToPostman(Mail *m, string name)*

Esta função adiciona a correspondência *m* ao vetor de correspondências que o carteiro de nome *name* é responsável por distribuir (*myMail*) e retorna esse carteiro. Se não existir um carteiro de nome *name*, a função deve lançar a exceção *NoPostmanException*. Esta classe exceção deve incluir o membro-função *getName()* que retorna o nome do carteiro não existente.