Parte prática. Duração: 2h00m

Uma cooperativa de produtores de vinhos deseja implementar um sistema de informação para gerir a sua produção. A cooperativa é formada por vinícolas que produzem vinhos, identificados por um nome e por um ano. Às vinícolas também podem estar associados enólogos, que são responsáveis pela qualidade dos vinhos por elas produzidos.

Para facilitar a consulta dos vinhos disponíveis na cooperativa, a classe **Cooperativa** guarda objetos da classe **Vinho** numa árvore de pesquisa binária (BST), organizados por ordem decrescente de ano; os vinhos de um mesmo ano são ordenados alfabeticamente. Os produtores da cooperativa são representados por objetos da classe **Vinicola**, e estão guardados na **Cooperativa** numa tabela de dispersão, indexados pelo nome da vinícola. Objetos da classe **Vinicola** também guardam informação do número de vinhos diferentes produzidos pela vinícola. Para efeitos de valorização do seu trabalho, a **Cooperativa** guarda os enólogos associados às vinícolas numa fila de prioridade. Um objeto da classe **Enologo** é identificado por um nome e mantém uma lista das vinícolas a que está associado. Um enólogo será tão mais produtivo quando maior o número total de vinhos das vinícolas que acompanha, destacando-se assim mais ao topo da fila de prioridade.

As classes **Cooperativa**, **Vinho**, **Vinicola** e **Enologo** estão parcialmente definidas a seguir. Para a correta utilização das respectivas estruturas de dados, note que poderá ser necessária a implementação de operadores e funções apropriadas.

Se desejar, poderá implementar funções e métodos auxiliares.

```
typedef tr1::unordered_set<Vinicola, hVinicola, hVinicola> hashVinicola;

class Cooperativa {
  BST<Vinho> vinhos;
  hashVinicola vinicolas;
  priority_queue<Enologo> enologos;
public:
  Cooperativa();
  void addVinhos(const vector<Vinho>& vv);
  list<string> vinhosDisponiveis(int ano1, int ano2);
  int delVinhoNome(string umVinho);
  void addVinicola(string umaVinicola);
  Vinicola maisOpcoes() const;
  void addEnologoVinicola(string umEnologo, string umaVinicola);
  list<Vinicola> vinicolasMelhoresNEnologos(int n);
};
```

```
class Vinho {
  string nome;
  int ano;
public:
  Vinho(string umNome, int umAno);
  string getNome() const;
  int getAno() const;
};
```

```
class Vinicola {
 string nome;
  int vinhos;
public:
  Vinicola(string umNome);
  string getNome() const;
  int getVinhos() const;
  void addVinho();
};
```

```
class Enologo {
      string nome;
      list<Vinicola> vinicolas;
public:
      Enologo(string umNome);
      string getNome() const;
      list<Vinicola> getVinicolas() const;
};
```

Parte prática. Duração: 2h00m

a1) [**2,5 valores**] Implemente o construtor da classe **Cooperativa**, a fim de inicializar corretamente a árvore de pesquisa binária. Implemente também o membro-função da classe **Cooperativa**:

void addVinhos(const vector<Vinho>& vv)

O método *addVinhos* recebe como argumento um vetor de vinhos que serão, um a um, inseridos na BST *vinhos*. Os vinhos estão guardados por ordem decrescente de ano; vinhos do mesmo ano são ordenados alfabeticamente.

a2) [**3,0 valores**] Implemente na classe **Cooperativa** o membro-função:

list<string> vinhosDisponiveis(int ano1, int ano2)

Este método retorna uma lista de *strings* com os nomes dos vinhos disponíveis e respetivos anos, separados por espaço, entre os anos *ano1* e *ano2*, inclusivé. Os nomes dos vinhos estão organizados em ordem alfabética inversa (decrescente), na lista retornada. Caso não existam vinhos disponíveis entre *ano1* e *ano2*, a lista resultante deverá conter uma única *string*: "Indisponivel".

a3) [**2,5 valores**] Implemente na classe **Cooperativa** o membro-função:

int delVinhoNome(string umVinho)

Esta função remove da BST *vinhos* aqueles de nome *umVinho*. Caso o mesmo vinho esteja disponível em diferentes anos, todas as suas edições, de anos distintos, serão também removidas da árvore. A função retorna o número de vinhos removidos da árvore, após a operação.

b1) [**3,5 valores**] Implemente na classe **Cooperativa** o membro-função:

void addVinicola(string umaVinicola)

Este método adiciona na tabela de dispersão *vinicolas*, uma nova vinícola de nome *umaVinicola*. As vinícolas são indexadas na tabela pelo seu *nome*. Ao ser inserida na tabela, uma vinícola tem seu membro-dado *vinhos* inicializado a 1. Caso a vinícola já exista na tabela, deve ser incrementado o membro dado *vinhos*, indicando que mais um vinho é disponibilizado por aquela vinícola. Uma vinícola é considerada igual a outra quando os seus nomes são iguais.

b2) [**2,5 valores**] Implemente na classe **Cooperativa** o membro-função:

Vinicola maisOpcoes() const

Este método retorna a *Vinicola* com o maior número de opções de vinhos disponíveis, ou seja, aquela com o membro-dado *vinhos* de maior valor. Se mais de uma vinícola oferecer o mesmo número de *vinhos*, o método retorna a de maior nome alfabeticamente. A vinícola deverá permanecer na tabela.

Parte prática. Duração: 2h00m

c1) [**3,5 valores**] Implemente na classe **Cooperativa** o membro-função:

void addEnologoVinicola(string umEnologo, string umaVinicola)

Este método associa uma vinícola a um enólogo, adicionando a vinícola de nome *umaVinicola* à lista *vinicolas* do enólogo de nome *umEnologo*. Considere que cada vinícola pode estar associada a apenas um enólogo. Caso o enólogo não exista, um novo enólogo é adicionado à fila de prioridade. No caso do enólogo já existir, a sua lista de vinícolas deve ser atualizada. No caso de o enólogo já ter na sua lista de vinícolas uma de nome *umaVinicola*, o membro-dado *vinhos*, da vinícola, deverá ser incrementado de uma unidade. Estas atualizações podem alterar a ordem do enólogo na fila de prioridade. O enólogo que somar o maior número de vinhos dentre todas as suas vinícolas, deve ficar à cabeça da fila.

c2) [**2,5 valores**] Implemente na classe **Cooperativa** o membro-função:

list<Vinicola> vinicolasMelhoresNEnologos(int n)

Este método retorna uma lista com as vinícolas dos *n* melhores enólogos da cooperativa, ou seja, aqueles com as *n* melhores somas de *vinhos* dentre as suas vinícolas, respectivamente. Cano *n* seja maior do que o número de enólogos na fila de prioridade, as vinícolas de todos os enólogos da cooperativa serão incluídas na lista resultante.