CI4 Parte prática. Duração: 2h00m

Numa clínica veterinária, pretende-se implementar um sistema de informação para gestão de consultas e dados relativos a animais e veterinários.

A informação sobre um **Animal** inclui o nome, espécie, idade e o nome do seu veterinário. Todos os animais possuem um veterinário e um veterinário trata vários animais. Considere que todos os animais possuem nomes diferentes.

A informação sobre um **Veterinario** inclui o nome, e uma lista dos nomes dos animais que trata na clínica.

A classe **Consulta** representa uma consulta e contém informação sobre a hora, dia e mês da consulta, bem como o nome do animal (doente).

A classe **ConsultaVet** contém informação sobre todas as consultas de um veterinário: inclui o nome do veterinário e uma árvore binária de pesquisa com as consultas do veterinário. Esta árvore está ordenada crescentemente por horário da consulta (hora, dia, mês).

A classe **Clinica** guarda:

- informação sobre as consultas disponíveis para todos os veterinários na lista *consultasDisponiveis* de objetos *ConsultasVet*.
- informação sobre os seus veterinários na fila de prioridade *veterinarios*. É política da clínica distribuir os animais pelos veterinários da clínica. Assim, a fila de prioridade *veterinarios*, deve conter no seu início o veterinário que possui menor número de animais a seu cargo e, no caso de empate, o de menor nome alfabeticamente.
- Informação sobre os animais na tabela de dispersão *animais*

As classes **Animal**, **Veterinario**, **Consulta** e **Clinica** estão parcialmente definidas a seguir.

```
typedef unordered_set<Animal, hAnimal, hAnimal> hashAnimal;

class Animal {
public:
  string especie;
  string nome;
  int idade;
  int numConsultas;
  string nomeVeterinario;
  Animal(string umaEspecie, string
umNome, int umaIdade);
};


class Veterinario {
  string nome;
  list<string> meusAnimais;
public:
  Veterinario(string umNome);
};

class ConsultasVet {
public:
   string nomeVeterinario;
   BST<Consulta> minhasConsultas;
}
```

```
class Consulta {
  int hora, int dia, int mes;
  string nomeAnimal;
public:
  Consulta(int umaHora, int umDia, int
umMes, string umAnimal);

};


class Clinica {
  hashAnimal animais;
  priority_queue<Veterinario> veterinarios;
  list<ConsultasVet> consultasDisponiveis;
public:
  Clinica();
};
```

a) Inicialize corretamente as árvores binárias de pesquisa e os operadores necessários à sua utilização.

a1) [3 valores] Implemente o membro-função da classe **Clinica**:

void addConsultas(const vector<Consulta> consultas1, string nomeVet)

Esta função adiciona os novos horários de consultas presentes no vetor `consultas1` às consultas disponíveis (`consultasDisponiveis`) do veterinário de nome `nomeVet`. Se o veterinário de nome `nomeVet` ainda não existe na lista de consultas disponíveis da clínica, deve ser criado um novo elemento relativo a este veterinário e adicionado no início da lista `consultasDisponiveis`.

a2) [2.5 valores] Implemente o membro-função da classe **Clinica**:

list<Consulta> veConsultasDisponiveis(int dia1, int dia2, int mesC, string nomeVet) const

Este método retorna uma lista das consultas disponíveis para o veterinário de nome `nomeVet`, entre os dias `dia1` (inclusivé) e `dia2` (inclusivé) do mês `mesC`. A lista resultante deve estar ordenada crescentemente por data de consulta.

a3) [2.5 valores] Implemente o membro-função da classe **Clinica**:

bool marcaConsulta(int &horaC, int &diaC, int &mesC, string nomeAnimal, string nomeVet)

O método `marcaConsulta` marca uma consulta para o animal de nome `nomeAnimal`, que possui o veterinário de nome `nomeVeterinario`, preferencialmente para o horário `horaC, diaC, mesC`. Se o veterinário não tiver disponibilidade neste horário, a consulta é marcada para o horário mais próximo (no futuro). Caso seja marcada com sucesso, o horário da consulta é retornado nos argumentos `horaC, diaC, mesC` e a função retorna `true`. Se o veterinário não tiver horário disponível na data indicada, ou após esta, a função retorna `false`. Se o veterinário não tiver qualquer horário (ou seja, não existir), a função também retorna `false`.

Nota: a marcação de uma consulta implica a eliminação dessa informação na lista `consultasDisponiveis`.

b) Codifique adequadamente os operadores necessários à utilização da tabela de dispersão

b1) [3 valores] Implemente o membro-função da classe **Clinica**:

Animal fimConsulta(string umNomeAnimal, string umNomeEspecie)

Esta função atualiza a informação presente na tabela de dispersão `animais`, no final de uma consulta do animal de nome `nomeAnimal`. Se o animal já existe na tabela, deve ser incrementado o número de consultas desse animal. Se o animal de nome nomeAnimal não existe na tabela, este deve ser adicionado à tabela com número de consultas igual a 1. A função retorna o objeto `Animal` atualizado ou inserido na tabela.

b2) [3 valores] Implemente o membro-função da classe **Clinica**:

int numAnimaisEspecie(string umNomeEspecie) const

Esta função retorna o número de animais da espécie `umNomeEspecie` presentes na tabela `animais`.

c) Codifique adequadamente os operadores necessários à utilização da fila de prioridade

c1) [**3 valores**] Implemente o membro-função da classe **Clinica**:

Veterinario alocaVeterinario(string umNomeAnimal)

Esta função escolhe um veterinário (da fila `veterinarios`) para tratar do animal de nome `umNomeAnimal`, tentando distribuir os animais pelos veterinários existentes. Assim, o veterinário escolhido é aquele que possui, neste momento, o menor número de animais a seu cargo. No caso de existirem dois veterinários com o mesmo número de animais, é escolhido o de menor nome, alfabeticamente. A função adiciona o animal à lista de animais do veterinário escolhido e retorna esse veterinário. Deve também atualizar a fila `veterinarios`.

c2) [**3 valores**] Implemente o membro-função da classe **Clinica**:

list<string> veterinariosMaisN(int n) const

Esta função retorna uma lista dos nomes dos veterinários que possuem mais de `n` animais a seu cargo. A lista deve conter o nome dos veterinários ordenados decrescentemente por número de animais a seu cargo.