MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO | 2o ANO

EIC0013 | ALGORITMOS E ESTRUTURAS DE DADOS | 2017-2018 – 1o SEMESTRE

CI2 Parte Prática. Duração: 2h00m

Nota: Submeta a sua resolução num ficheiro *zip*

O Hospital da Vida tem vários médicos de várias especialidades. Os médicos estão armazenados na lista

doctors. A cada médico está associado um código (codeM), uma especialidade (medicalSpecialty) e

uma fila de doentes (patient) que aguardam consulta. Cada doente é identificado por código (codeP) e

pela especialidade que procura. Os doentes para cada especialidade só podem ser atendidos por médicos

dessa especialidade.

Quando solicita uma consulta, o doente é colocado na fila de um médico da especialidade que procura. Após

a consulta, o ficheiro do doente (considera-se, neste caso que o ficheiro do doente é o próprio doente) é

colocado na bandeja dos doentes consultados (lista letterTray), um por cima do outro, para

processamento pelo departamento administrativo do Hospital. A capacidade de uma bandeja é o número

máximo de ficheiros que esta pode conter.

Considere as classes **Patient**, **Doctor** e **Hospital**, que estão parcialmente definidas a seguir:

class Patient {

unsigned codeP; string medicalSpecialty; public:

Patient(unsigned codP, string mS); // ... }; class Doctor {

unsigned codeM; string medicalSpecialty; queue<Patient> patients; public:

Doctor(unsigned codM, string mS); Doctor(unsigned codM, string mS, queue<Patient> patients1);

// ... };

class Hospital {

list<Doctor> doctors; list<stack<Patient> > letterTray; unsigned trayCapacity; public:

Hospital(unsigned trayC=20); // ... }; // ... };

ANA PAULA ROCHA, HENRIQUE LOPES CARDOSO, CARLOS SOARES, LILIANA FERREIRA 22/NOVEMBRO/2017 **|** PÁG. 1 / 3

**a)** Implemente na classe **Hospital** o membro-função:

*unsigned numPatients(string medicalSpecialty) const*

Esta função retorna o número de doentes que aguardam consulta da especialidade medicalSpecialty

no Hospital, isto é, que estão presentes na lista doctors. Note que pode existir mais do que um médico

da mesma especialidade.

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO | 2o ANO

EIC0013 | ALGORITMOS E ESTRUTURAS DE DADOS | 2017-2018 – 1o SEMESTRE

**b)** Implemente na classe **Hospital** o membro-função:

*void sortDoctors()*

que ordena a lista de médicos (doctors) por ordem crescente de ocupação (número de doentes a

aguardar consulta). Em caso de empate, a ordenação é realizada por ordem alfabética da especialidade

do médico (medicalSpecialty). Em caso de empate, a ordenação é realizada por ordem crescente

de código.

**c)** Implemente na classe **Doctor** o membro-função:

*void moveToFront(unsigned codP1)*

que, por necessidade urgente de um doente de código codP1, coloca este doente no início da fila

patients. Os restantes doentes presentes na fila mantêm a sua posição relativa. Se não existir nenhum

doente de código codP1 na fila patients, esta mantém-se inalterada.

**d)** Implemente na classe **Hospital** o membro-função

*bool addDoctor(unsigned codM1, string medicalSpecialty1)*

que insere um novo médico de código codM1 e especialidade medicalSpecialty1 no final da lista

doctors, se o número de médicos dessa especialidade existente no hospital for menor que 3, caso em

que a função retorna true. Se o número de médicos da especialidade for maior ou igual a 3, o médico

não é adicionado à lista e a função retorna false.

**e)** Implemente na classe **Hospital** o membro-função:

*queue<Patient> removeDoctor(unsigned codM1)*

Esta função elimina da lista doctors o médico de código codM1 e retorna a fila de doentes desse

médico. Se o médico não existir, deve ser lançada a exceção **DoctorInexistent** (esta exceção já está

definida na classe **Hospital**).

ANA PAULA ROCHA, HENRIQUE LOPES CARDOSO, CARLOS SOARES, LILIANA FERREIRA 22/NOVEMBRO/2017 **|** PÁG. 2 / 3

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO | 2o ANO

EIC0013 | ALGORITMOS E ESTRUTURAS DE DADOS | 2017-2018 – 1o SEMESTRE

**f)** Implemente na classe **Hospital** o membro-função:

*bool putInLessBusyDoctor(unsigned codP1, string medicalSpecialty1)*

Esta função aceita o doente de código codP1 e consulta de especialidade medicalSpecialty1 para

consulta no hospital. O doente deve ser adicionado à fila menos ocupada (com menor número de

elementos) dos médicos da especialidade medicalSpecialty1.

Se não existir nenhum médico com a especialidade pretendida, a função retorna false. Caso contrário,

retorna true.

Nota: Na classe **Doctor**, já está implementado o membro-função void addPatient(const Patient

&p1), que adiciona o doente p1 no final da lista de doentes do médico.

**g)** Implemente na classe **Hospital** o membro-função:

*void processPatient(unsigned codM1)*

Esta função termina a consulta do próximo doente na fila do médico com o código codM1 da lista

doctors. O doente é retirado da fila e colocado na primeira bandeja disponível na lista letterTray.

As bandejas possuem uma capacidade limitada, pelo que só pode colocar o doente na bandeja que ainda

tiver espaço livre. Se não existir nenhuma bandeja com espaço livre, deve criar uma bandeja nova

(pilha) e colocá-la no final da lista letterTray.

Se não existir o médico de código codM1 ou a sua fila de doentes estiver vazia, nada é alterado.

Nota: Na classe **Doctor**, já está implementado o membro-função Patient removeNextPatient(),

que retira o próximo doente da fila de doentes do médico e retorna esse doente (lança uma exceção se

a fila está vazia).

**h)** Implemente na classe **Hospital** o membro-função:

*unsigned removePatients(unsigned codP1)*

Esta função retira os doentes de código codP1 existentes no topo (e apenas no topo) das bandejas da

lista letterTray. Se alguma bandeja ficar vazia, esta deve ser eliminada da lista. Retorna o número

de doentes que retirou.

ANA PAULA ROCHA, HENRIQUE LOPES CARDOSO, CARLOS SOARES, LILIANA FERREIRA 22/NOVEMBRO/2017 **|** PÁG. 3 / 3