Aluno(a):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Matrícula:\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| 1a) | 2.5 |  |
| 2a) | 2.5 |  |
| 3a) | 2.5 |  |
| 4a) | 2.5 |  |
|  | 10.0 |  |

1. A prova é individual e sem consulta.
   1. **Qualquer tentativa de “cola” resultará na anulação da prova do aluno ou dos alunos envolvidos**.
   2. Os aparelhos celulares deverão permanecer desligados durante toda a prova. **Aparelhos celulares ligados serão tratados como tentativa de “cola”**.
2. A interpretação faz parte da questão.
   1. **Não há perguntas durante a prova.**
   2. Em caso de dúvida escreva a dúvida e a sua interpretação na resposta.
3. **Após o início da prova, não será possível sair e voltar à sala.**
4. As respostas devem seguir as questões. Caso precise de rascunho use o verso da folha.
5. A prova pode ser feita a lápis.
6. O tempo de prova é 1:45 h.

**Questão 1** (2.5 pontos) Considere um banco cujas agências são identificadas por um código de 5 dígitos, onde o último dígito, chamado de *dígito verificador*, é computado a partir dos 4 primeiros e serve para detectar códigos incorretos. Os códigos das agências não são atribuídos sequencialmente.

Assuma que os códigos são representados como *strings* de comprimento 5 e que são armazenados em uma tabela de *hash*.

a) (0.5 ponto) Suponha que o banco possua 400 agências. Qual deve ser o tamanho, *tam*, da tabela de *hash*? Explique sua resposta.

b) (1.0 ponto) Escreva em C uma função de dispersão (uma função *hash*) que recebe como argumentos:

* um *string* de 5 dígitos, *cod*, representando o código de uma agência
* um inteiro *tam* que define o tamanho da tabela *hash*

e retorne um inteiro que identifica a entrada na tabela de *hash* para a qual *cod* é mapeado.

c) (0.5 ponto) Argumente porque a sua implementação está correta e representa uma função de hash adequada.

d) (0.5 ponto) Suponha que o banco expanda as suas operações e abra 100 novas agências. Qual será o efeito desta expansão e o que deve ser modificado na função de hash e na tabela de hash para acomodar esta expansão? Explique a sua resposta.

Resposta:

**Questão 2** (2.5 pontos) Considere o problema de representar 2 subconjuntos de um grupo *G* de pessoas, definidos como:

*G1* = { *x* ∈ *G* / *x* estuda Engenharia de Computação }

*G2* = { *x* ∈ *G* / *x* é já cursou INF1010 }

Suponha que os subconjuntos estão representados utilizando vetores de bits:

struct \_bitvector {

int max;

int \*vector;

};

typedef struct \_bitvector BitVector;

a) (1.5 pontos) Implemente em C uma rotina que receba como entrada *G1* e *G2* e devolva o subconjunto *H* de *G* definido como:

*H* = { *x* ∈ *G* / *x* estuda Engenharia de Computação e *x* ainda não cursou INF1010 }

A rotina não deverá chamar as operações definidas em sala para a estrutura escolhida.

b) (0.5 ponto) Qual a complexidade, no pior caso, da rotina implementada no item (a), medida em termos do número de elementos de cada subconjunto. Explique sua resposta.

c) (0.5 ponto) Baseando-se na resposta do item (b), argumente porque a sua implementação é mais eficiente do que uma implementação da rotina do item (a) que utilize listas encadeadas para representar os conjuntos.

Resposta:

**Questão 3** (2.5 pontos) Considere o problema de agrupar os empregados de uma empresa pelo departamento em que trabalham, assumindo que um empregado só pode trabalhar em um departamento e que um empregado pode mudar de departamento ao longo do tempo. Assuma ainda que se deseja realizar duas operações:

* Determinar eficientemente se dois empregados trabalham no mesmo departamento;
* Implementar eficientemente a fusão de 2 departamentos em um só.

a) (1.0 ponto) Qual estrutura de dados, dentre as apresentadas em sala, seria mais eficiente para resolver o problema acima? Explique sua resposta, indicando como as operações acima serão implementadas.

b) (1.5 pontos) Defina em C a estrutura de dados escolhida no item (a) e implemente uma rotina para determinar, da forma mais eficiente possível, se dois empregados não trabalham no mesmo departamento. A rotina não deverá chamar as operações definidas em sala para a estrutura escolhida.

Resposta:

**Questão 4** (2.5 pontos) Considere o problema de organizar uma fila de atendimento em que as pessoas mais velhas tenham prioridade sobre as mais novas. Caso duas pessoas tenham a mesma idade, a ordem é indistinta. Assuma ainda que se deseja realizar as seguintes operações básicas:

* Organizar inicialmente um conjunto de pessoas, segundo a prioridade por idade;
* Atender, ou seja, tirar da fila de atendimento, a pessoa de maior idade;
* Incluir uma nova pessoa na fila de atendimento, segundo a prioridade por idade.

a) (0.5 ponto) Porque um *heap* seria a estrutura mais eficiente para resolver o problema acima? Explique sua resposta.

b) (0.5 ponto) Considere inicialmente um conjunto de pessoas com as seguintes idades: 15, 60, 30, 45, 90. Mostre como construir o *heap*, utilizando um vetor, da forma mais eficiente possível, para organizar este conjunto de pessoas na fila de atendimento.

c) (1.5 ponto) Escreva em C uma função que recebe como argumento um *heap*, organizado como nos itens anteriores, e um inteiro *i* e remove o elemento do *heap* que ocupa a *i*-ésima posição do vetor. A rotina deve retornar 1 se a alteração ocorreu corretamente e 0 em caso contrário. Assuma que o *heap* seja definida como abaixo:

struct \_heap {

int max; /\* tamanho máximo do heap \*/

int pos; /\* próxima posicao disponível no vetor \*/

float\* prioridade; /\* vetor das prioridades \*/

};

typedef struct \_heap Heap;

Assuma ainda que a assinatura da rotina seja:

int heap\_altera(Heap\* heap, int i)

Resposta: