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

|  |  |  |
| --- | --- | --- |
| 1a) | 2,5 |  |
| 2a) | 2,5 |  |
| 3a) | 2,5 |  |
| 4a) | 2,5 |  |
|  | 10,0 |  |

**LEIA COM CUIDADO**

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 e guardados fora do alcance durante toda a prova. **Aparelhos celulares ligados ou de alguma forma visíveis 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. O tempo de prova é 1:45 h.
4. **Após o início da prova, não será possível sair e voltar à sala.**
5. As respostas devem seguir as questões. Caso precise de rascunho use o verso da folha.
6. A prova pode ser feita a lápis.

**Questão 1 (2,5 pontos)**.

**(a) (1,5 pontos)** Escreva uma rotina em C para inserção de chaves, definidas como números inteiros, em uma tabela de dispersão com endereçamento aberto e função de dispersão *h* definida como

onde

*K* = chave a ser inserida

*i* = 0, 1, 2, ... = número da tentativa de inserção da chave na tabela

*P* = constante para redispersão, em caso de colisão

*N* = tamanho da tabela de dispersão

Assuma que as posições são preenchidas, inicialmente, com o valor -1 e que a tabela seja declarada como:

#define N 24679; /\* tamanho da tabela de dispersão \*/

#define P 23; /\* coeficiente para redispersão \*/

#define VAZIO (-1); /\* indica que a posição está desocupada \*/

int tabela[N]; /\* tabela de dispersão \*/

A função deve ter o seguinte protótipo:

int insere(int K)

e deve retornar:

-1 se a chave não puder ser inserida

0 se a chave já existe

1 se a chave for inserida corretamente

**(b) (0,5 ponto)** A escolha de *N* = 24.679 é uma boa escolha, supondo-se que o número de chaves não ultrapasse 22.000 chaves? Explique a sua resposta cuidadosamente.

**(c) (0,5 ponto)** A função de dispersão adotada é adequada para as escolhas de *N*=24.679 e *P*=23? Explique a sua resposta cuidadosamente.

**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* é gestante }

*G2* = { *x* ∈ *G* / *x* vive em área infestada de “Aedes aegypti” }

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* é gestante e *x* não vive em área infestada de “Aedes aegypti” }

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 a árvore B de ordem 5 (todo nó, exceto a raiz, tem nó mínimo 2 chaves e no máximo 4 chaves) da Figura 1.

100

53 77 123 200

10 40 60 70 80 90 110 113 130 138 140 170 230 236 242 245

**Figura 1**. Árvore-B da Questão 3.

Realize as seguintes operações, **utilizando sempre a árvore da Figura 1 como árvore inicial**. Indique os nós que sofrem modificações após cada operação, bem como a ocorrência de divisão, redistribuição ou concatenação:

1. **(0,5 ponto)** Inserção de 142.
2. **(1,0 ponto)** Remoção de 60.
3. **(1,0 ponto)** Remoção de 123 de duas maneiras diferentes.

**Resposta**

**Questão 4 (2,5 pontos)**. Considere uma grade não regular tridimensional tal que as fronteiras lateral esquerda, inferior e frontal não pertencem à grade; para cada célula, as fronteiras lateral esquerda, inferior e frontal não pertencem à célula. Cada célula da grade armazena uma lista encadeada de pontos.

Considere os seguintes tipos que representam a grade:

typedef struct lista Lista;

struct lista {

float x, y, z; /\* ponto na lista \*/

Lista\* prox; /\* ponteiro para próximo elemento da lista \*/

};

typedef struct grade Grade;

#define NX 200 /\* número de elementos do vetor gX \*/

#define NY 135 /\* número de elementos do vetor gY \*/

#define NZ 400 /\* número de elementos do vetor gZ \*/

struct grade {

float gX[NX]; /\* vetor que define as divisões no eixo X da grade \*/

float gY[NY]; /\* vetor que define as divisões no eixo Y da grade \*/

float gZ[NZ]; /\* vetor que define as divisões no eixo Z da grade \*/

Lista\* prim[NX][NY][NZ]; /\* lista por célula (inicializada com NULL) \*/

};

Implemente uma função que receba como entrada uma grade *g*, com a definição acima, e um ponto *(px, py, pz)*, remove o ponto da grade, se existir, e retorne

0 se o ponto não existia na grade

1 se o ponto existia e foi removido da grade

-1 se o ponto está fora da grade

A função deve seguir o seguinte protótipo:

int remove(Grade\* g, float x, float y, float z);

**Resposta**