Algoritmos e Estrutura de Dados II – 2012/1 – Lista III-B (0.5 pontos)

Nome:

Assinatura:

---

**Questão 1.** Considere as seguintes estruturas de dados.

```
struct numero_conta {
    int agencia;
    int conta;
};

struct cliente {
    char nome[128];
    int cpf;
    struct numero_conta conta;
    int telefone;
    int cep;
    char endereco[1024];
};
```

Implemente três funções de pesquisa binária como a seguir:

```
int binsearch_cpf(struct cliente *v, int cpf);
int binsearch_nome(struct cliente *v, char *nome);
int binsearch_conta(struct cliente *v, struct numero_conta conta);
```

Estas funções devem fazer uma busca binária no vetor `v` pelas chaves recebidas como parâmetro. Assuma que o vetor `v` está ordenado pela chave usada na busca. Suas funções devem retornar o índice do cliente no vetor `v` ou `-1` se não existir cliente com a chave procurada.

---

**Questão 2.** Modifique as funções implementadas na Questão 1 de forma que elas retornem o índice do cliente com a maior chave menor ou igual à chave procurada. Assuma que não existem clientes com chave repetida. Nota: O índice retornado é o índice onde o cliente seria inserido para manter o vetor `v` ordenado.

---

**Questão 3.** Defina um tipo abstrato de dados `struct conjunto` e implemente as seguintes funções:

```
/* criar um novo tipo abstrato de dados conjunto */
struct conjunto * cria_conjunto(void);

/* inserir o registro [reg] no conjunto [c] */
void insere_conjunto(struct conjunto *c, struct registro *reg);

/* pesquisa pela chave contida no registro [reg]. retorne um
 * ponteiro para o registro contido em [c] que possui a mesma chave
 * que [reg]. se nao houver registro em [c] com a mesma chave que
 * [reg], retorne NULL. */
struct registro * pesquisa_conjunto(struct conjunto *c, struct registro *reg);

/* remove o registro em [c] que possui a mesma chave que [reg] e
 * retorna um ponteiro para esse registro. se nao houver registro
 * em [c] com a mesma chave que [reg], retorne NULL. */
struct registro * remove_conjunto(struct conjunto *c, struct registro *reg);
```

**Questão 4.** Dada a implementação do tipo abstrato de dados na Questão 3, indique quantas comparações serão feitas para inserir as seguintes chaves: `q u e s t a o f c i l`.

---

**Questão 5.** Implemente uma função de busca sequencial em lista encadeada (ponteiros). Transforme essa função numa função auto-organizante: toda vez que uma busca com sucesso for realizada, coloque o elemento buscado no começo da lista encadeada. Essa heurística tenta colocar os elementos mais populares no começo da lista para reduzir o tempo médio em uma busca com sucesso.

---

**Questão 6.** Considere um procedimento de criação de lista auto-organizante como definida na Questão 5 com dois passos. Primeiro, fazemos uma busca por um elemento que queremos inserir (possivelmente movendo-o para o começo da lista). Se o elemento não estiver presente, inserimos o elemento no início da lista. Mostre a ordem final da lista auto-organizante depois da inserção dos seguintes elementos: `q u e s t a o f a c i l`.

---

**Questão 7.** Implemente uma função de busca binária não-recursiva.

---

**Questão 8.** Modifique uma função de busca binária de forma que ela retorne o número de elementos com a chave procurada. (Não é preciso retornar o índice dos elementos.)

---

**Questão 9.** Encontre os tamanhos (aproximados) de vetores de entrada para os quais busca binária é 10, 100 e 1000 vezes mais rápida que busca sequencial.

---

**Questão 10.** Telefones em Belo Hozizonte começam com código 31 e têm 8 números. Implemente um tipo abstrato de dados que permita saber se um número de telefone já existe em $O(1)$. Quando novos telefones são vendidos ou contratos cancelados, seu tipo abstrato de dados deve ser capaz de inserir e remover telefones na base de dados com custo $O(1)$. Seu tipo abstrato de dados deve utilizar no máximo 12500000 bytes de memória (aproximadamente 12 MiB) para qualquer número de telefones na base de dados.

---

**Questão 11.** Considerando as estruturas abaixo, implemente uma função *não-recursiva* de busca em árvore binária de pesquisa. Sua função deve seguir o cabeçalho abaixo.

```
struct arvore {
    struct arvore *esq;
    struct arvore *dir;
    struct registro *reg;
};

struct registro {
    int chave;
    /* outros dados. */
};

struct registro * busca_iterativa(struct arvore *a, int chave);
```

---

**Questão 12.** Modifique a resposta da Questão 11 para implementar uma função não-recursiva de inserção em árvore binária de busca.

**Questão 13.** Mostre a árvore binária de pesquisa resultante da inserção dos elementos `q u e s t a o f a c i l` pela função implementada na Questão 12. Mostre também o total de comparações feitas durante o processo.

---

**Questão 14.** Se soubermos a frequência de busca por cada elemento em uma base de dados, devemos começar a incluí-los numa árvore de busca pelos mais ou menos frequentemente buscados? Justifique sua resposta.

---

**Questão 15.** Desenhe a árvore SBB resultante da inserção das chaves `q u e s t a o f a c i l`. Assuma que a árvore não contém chaves duplicadas.

---

**Questão 16.** Qual é a altura máxima e mínima de uma árvore SBB com $N$ folhas (ou seja, $N$ registros)?

---

**Questão 17.** Implemente uma função de *hashing* para cadeias de caracteres com o seguinte cabeçalho: `int hash(char *chave)`.

---

**Questão 18.** Mostre o estado final de um dicionário com resolução de conflitos por encadeamento depois da inserção das chaves `q u e s t a o f a c i l`. Considere que o dicionário não armazena chaves duplicadas, que o valor de cada caractere é sua ordem no alfabeto e que o dicionário possui uma tabela com 5 listas (isto é, $M = 5$).

---

**Questão 19.** Repita a Questão 18 para um dicionário onde as listas encadeadas em cada posição da tabela são mantidas ordenadas. O custo de inserção dos elementos depende da ordem de inserção? Justifique sua resposta.

---

**Questão 20.** Quanto tempo pode levar a inserção de $N$ elementos num dicionário com resolução de conflitos com endereçamento aberto? Justifique sua resposta.

---

**Questão 21.** Mostre o conteúdo de um dicionário com resolução de conflitos com endereçamento aberto depois da inserção das chaves `q u e s t a o f c i l`. Considere que a tabela do dicionário tem 17 posições e que o valor de cada caractere é sua ordem no alfabeto.

---