3ª Prova de Algoritmos e Estruturas de Dados I - Curso: Engenharia de Computação

Nome do aluno: _____ **Data:** 31/10/2012

Resolva as questões abaixo identificando-as claramente na folha de respostas. Mantenha o silêncio na sala (mantendo desligado aparelhos eletrônicos). A interpretação das questões faz parte da prova. **Horário limite:** 15:50

(10 pontos) 1) O que é algoritmo?

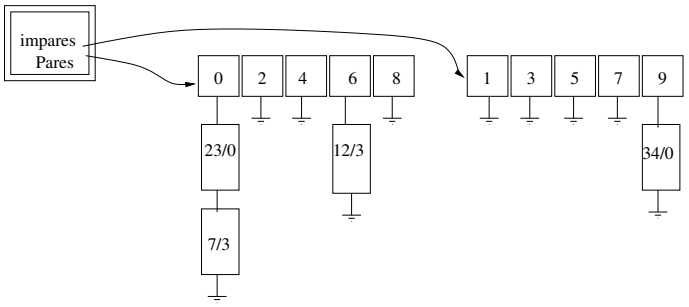
(20 pontos) 2) Sobre Tabelas HASH:

a) É possível criar uma função HASH em que é garantido que nunca haverá colisão? Porquê?

a) Quando existe colisão quais estratégias podem ser utilizadas para contornar esse problema? Quando cada estratégia é melhor utilizada?

(40 pontos) 3) Um aluno atrapalhado de estruturas de dados decidiu criar a sua própria variação de uma matriz esparsa. Ele decidiu alterar a estratégia de representar um vetor coluna e listas encadeadas para cada posição deste vetor por uma estratégia que utiliza 2 vetores colunas, uma para representar as colunas pares e outro para representar as colunas ímpares, tal como na figura. Ajude o tal estudante na sua implementação, criando as funções `cria_matriz`, `insere_elemento`, `consulta_elemento`

PS: Lembre-se que na função insere, você deve verificar se já existe o elemento na matriz e neste caso apenas atualizar o seu valor. Você pode achar conveniente escrever uma função `pesquisa`.

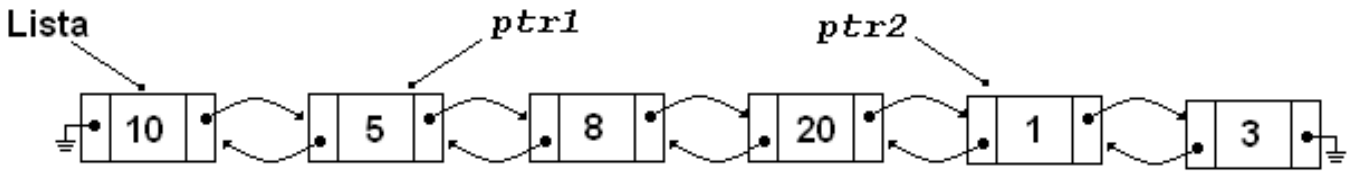



```c
struct tipo_elemento {
   int valor;
   int coluna;
   struct tipo_elemento *proximo;
};
// define o tipo lista
typedef struct {
   int quantidade;
   struct tipo_elemento *inicio;
}Tipo_Lista;

// define o tipo matriz
typedef struct {
   int mat_colunas;
   int mat_linhas;
   Tipo_Lista *Impar;
   Tipo_Lista *Par;
}Tipo_Matriz;
```

(30 pontos) 4) Seja uma lista duplamente encadeada, com a estrutura e exemplo abaixos:

```
typedef struct _nodo {
    struct _nodo *anterior;
    struct _nodo *proximo;
    int elemento;
} TipoNodo;
```



Sejam dados dois ponteiros para dois elementos distintos (ptr1 e ptr2). Assumindo que:

- Estes ponteiros existem e não serão passados valores nulos;
- Que podem estar em qualquer posição da lista;
- ptr1 vem "antes" na lista, de ptr2;
- Pode existir qualquer quantidade de elementos antes de ptr1, depois de ptr2 e entre ptr1 e ptr2.

Escreva uma função em C que elimina todos os elementos entre ptr1 e ptr2 (inclusive).