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

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. **Data:** 01/07/2016
Horário limite: 15:40

Perguntas comuns e suas respostas:

- P: Tenho uma dúvida na questão tal.
  R: A compreensão do enunciado faz parte da prova.
- P: O que será corrigido?
  R: A lógica, a criatividade, a sintaxe, o uso correto dos comandos e dos tipos, os nomes das variáveis, a indentação, uso equilibrado de comentários no código e, evidentemente, a clareza. Nesta prova, você deverá sobretudo escrever códigos modulares, usando corretamente funções e/ou procedimentos, conforme o caso, além de uso correto de variáveis locais ou globais e a passagem de parâmetros por referência ou por valor.
- P: Posso fazer a lápis?
  R: Não. A prova deverá ser feita a caneta.
- P: Posso responder na folha de questões?
  R: Não. A prova deverá ser respondida na folha de respostas.

(10 pontos) 1) O que é Algoritmo?

(5 pontos) 2) Na linguagem C temos a função `malloc()`, fornecida pela `stdlib.h`. Qual o uso desta função? O que a função retorna? Mostre um exemplo de uso desta função e explique todos os passos.

(5 pontos) 3) Diferencie passagem de parâmetro por referência e por cópia. Mostre um exemplo de cada.

(15 pontos) 4) Sobre algoritmos de ordenação:

a) É possível afirmar que o algoritmo QuickSort é superior aos algoritmos elementares (ex: BubbleSort, InsertionSort, SelectionSort) em todos os casos? Porquê?

b) Explique alguma estratégia para minimizar o problema apresentado no item (a).

c) A respeito da estabilidade de um algoritmo de ordenação, defina o conceito de estabilidade e cite 2 algoritmos estáveis e 2 não estáveis.

(35 pontos) 5) Um animado aluno está criando um sistema centralizado que recebe submissões de problemas, no estilo do MOJ, porém ele recebe submissões de diversas regiões do país.

O interessante é que o sistema processa as submissões a cada 5 minutos e por isso recebe muitos dados. Cada região envia uma lista encadeada, ordenada pelo tempo decorrido da prova em que a submissão foi feitas, ou seja, se uma submissão foi feita após 5 minutos de prova ela terá valor 5, se for 15 minutos, será 15.

O sistema deverá processar todas submissões, de todas as regiões. A ordem de processamento das submissões deverá ser ordenada pelo tempo decorrido da prova, ou seja, a submissão mais antiga (menor valor de tempo decorrido da prova), independente da região, é a que deverá ser feita primeiro.

O problema é relativamente simples, pois como todas as listas recebidas já estão ordenadas, basta criar um algoritmo que junte (intercale) todas as listas encadeadas em uma única lista, porém o seu algoritmo **NÃO** deve alocar novas células de memória.

Foi pedido a você que implemente 2 funções, uma chamada `intercala4` e outra `intercala8`, estas funções recebem, respectivamente, 4 e 8 ponteiros do tipo `struct no`, sendo, cada um, o início de uma lista encadeada ordenada pelo `tempo_de_submissao` e deverá retornar um único ponteiro do tipo `struct no` contendo uma lista resultante da intercalação das listas recebidas como parâmetro. Você deverá implementar da forma mais eficiente possível.

Dica: Você pode achar conveniente implementar a função `intercala2`.

A struct da lista encadeada é:

```c
struct no {
    int tempo_de_submissao;
    struct codigo;
    struct no *proximo;
};
```

(30 pontos) 6) Matriz esparsa

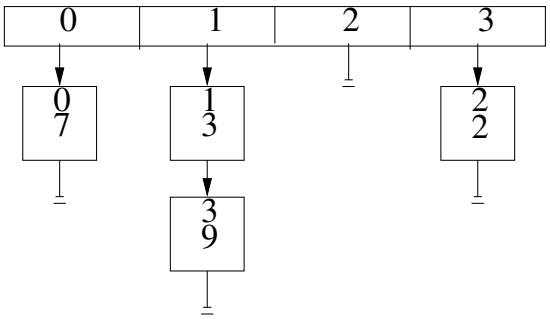
A estratégia clássica de representar uma matriz esparsa é com um vetor de colunas, e em cada índice deste vetor temos uma lista encadeada composta pelos elementos não nulos da coluna.

Observe na tabela 1 uma representação padrão de matriz, representada em um grid MxN, no caso 4x4. Na tabela 1 temos uma matriz esparsa, ou seja, a maior parte de seus elementos são nulos. Agora observe que a figura 1 representa esta matriz utilizando a estratégia clássica, comentada acima.

Tabela 1: Matriz Esparsa em uma representação padrão de Matriz

	0	1	2	3
0	7	0	0	0
1	0	3	0	0
2	0	0	0	2
3	0	9	0	0

Figura 1: Representação de Matriz esparsa com vetor de colunas e lista encadeada. Esta é a representação da matriz acima.



Considerando que uma matriz esparsa armazena números inteiros de 64bits e que os ponteiros também possuem 64bits e a representação clássica, responda:

a) (10 pontos) Quando esta representação deixa de ser vantajosa quando comparada com uma representação padrão de matriz? Explique.

b) (15 pontos) Implemente a função `insere` que recebe como parâmetro o vetor de colunas da matriz, a linha, a coluna e o elemento da matriz e armazene na representação de matriz esparsa.

c) (5 pontos) Escreva uma função `grau` que recebe como parâmetro o vetor de colunas da matriz e imprima quantos elementos não nulos existe em cada coluna.