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:** 07/12/2016
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.

(05 pontos) 1) O que é Algoritmo?

(05 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.

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

(05 pontos) 4) É possível afirmar que o algoritmo do QuickSort é mais eficiente que o algoritmo do MergeSort sempre? Explique.

(15 pontos) 5) Durante as aulas de estruturas de dados foram estudados alguns algoritmos de ordenação, entre estes, o método de ordenação por seleção. Quando estes métodos foram estudados, utilizou-se vetores para implementá-los. É claro que o método pode ser adaptado para ordenar elementos numa lista encadeada. Assim, pede-se que você implemente a função `ordenada(Tipo_Lista *L)` que ordena os elementos de uma lista encadeada utilizando o método de **seleção**. Note que você deve adaptar o método para funcionar como um lista encadeada e **não** copiar os elementos para um vetor auxiliar e ordená-los. Considere que as funções inicializa e insere estão disponíveis e os elementos já foram inseridos.

(05 pontos) 6) Um novo aluno dos Pampas está com problemas ao desenvolver um programa em C que possui uma lista simplesmente encadeada. Por algum motivo a sua função insere não está de fato inserindo o elemento na lista. Você é a única pessoa que pode ajudá-lo com esse problema.

O código da função insere está abaixo. Fale sobre o problema e como resolvê-lo.

```
1  struct celula {
2      int conteudo;
3      struct celula *prox;
4  };
5
6  struct TipoLista {
7      struct celula *inicio;
8      int quantidade;
9  };
10
11 void insere-inicio( int x, struct TipoLista *l)
12 {
13   struct celula nova;
14   nova.conteudo = x;
15   nova.prox = l->inicio;
16   l->inicio = &nova;
17   (l->quantidade)++;
18 }
```

(10 pontos) 7) É possível implementar, eficientemente, uma Fila utilizando uma lista simplesmente encadeada? Mostre qual é a função crítica para se implementar e como contornar o problema, caso exista.

(20 pontos) 8) O problema da intercalação (merge) é clássico na computação e consiste em resolver o seguinte problema: dados vetores crescentes `v[p .. q-1]` e `v[q .. r-1]`, rearranjar `v[p .. r-1]` em ordem crescente.

Uma *duoleO* é uma lista duplamente encadeada que contém uma sequência crescente de números inteiros. Escreva uma função que intercale duas *duoleO* dadas, produzindo assim uma terceira *duoleO*. Sua função **não** deve alocar novas células na memória, mas reaproveitar as células das duas listas dadas.

(30 pontos) 9) Matriz esparsa

Uma matriz esparsa A com t elementos não nulos pode ser armazenada em uma matriz $E_{t\times 3}$ da seguinte forma: As posições $E[0,0], E[0,1]$ e $E[0,2]$ armazenam o número de linhas, número de colunas e o número de elementos não nulos da matriz A. Os elementos $E[i,0], E[i,1]$ e $E[i,2]$, $i = 1, \ldots, t$ armazenam os índices de linha, coluna e o valor dos elementos não nulos respectivamente.

a) (05 pontos) Mostre a matriz abaixo utilizando a representação acima.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 23 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 42 & 0 & 0 & 0 & 0 \\ 0 & 8 & 0 & 16 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 \end{bmatrix}$$

b) (15 pontos) Implemente uma função que leia uma matriz, pela entrada padrão, e a armazene utilizando a estratégia definida acima, assuma que o seu programa recebe nos argumentos o tamanho da matriz.

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