Universidade Federal do ABC	**Avaliação**: Prova 01
Disciplina: Algoritmos e Estruturas de Dados I	**Turma**: Noturno
Professor: Jesús P. Mena-Chalco	**Data**: 22/03/2016

Nome completo:		RA:

Instruções para a prova (leia antes de começar):

1) A prova tem a duração de 1h50min.
2) É proibido o uso de qualquer aparelho ou recurso de processamento e/ou comunicação.

Questão 1 (2 pontos):
Qual o valor da variável x após a execução destas operações:

```
int x;
int v[] = {1,2,3,4,5,6,7};

x = *(v + *(v+1) + *(v+3));
```

X = 7

Questão 2 (6 pontos):
Escreva uma função recursiva que, dado um número inteiro positivo, devolva o maior digito. Por exemplo, para o número 37406 o valor devolvido será 7.

```
int maiorDigito(int n) {
  if (n<=0) {
    return 0;
  }
  else {
    int maior = maiorDigito(n/10);

    if (n%10>maior)
      maior = n%10;

    return maior;
  }
}
```

Questão 3 (2,5 pontos):
Seja o vetor H[0,…,n-1] um vetor associado a um Max-Heap. Quals as posições onde o segundo maior elemento do vetor H possa ser encontrado?. Justifique

- Se as posições do vetor iniciarem em 0, o segundo elemento maior estará ou na posição H[1] ou H[2].
- Se as posições do vetor iniciarem em 1, o segundo elemento maior estará ou na posição H[2] ou H[3].

dado que, em um Heap Máximo qualquer vértice (ou nodo) pai é maior ou igual ao seus filhos (esquerdo e direito).

Questão 4 (4 pontos):
Uma empresa deseja comprar um software para resolver sistemas. Existem no mercado **seis** programas para tal tarefa com custos operacionais diferentes. Seja n o tamanho da entrada para o sistema, o tempo gasto para cada sistema é:

$$S_1 = 2n \qquad S_4 = n\log(n^2)$$
$$S_2 = 10n \qquad S_5 = n\log(\sqrt{n})$$
$$S_3 = 100n \qquad S_6 = n\log(n)$$

Do ponto de vista de **análise assintótica**, qual (quais) sistema(s) **não** deveria(m) ser escolhido(s)? Justifique sua resposta.

Os sistemas S4, S5, e S6 não deveriam ser escolhidos pois, do ponto de vista de análise assintótico, todos eles tem um custo computacional proporcional a n log(n).

Os Sistemas S1, S2 e S3 tem um custo computacional proporcional a n.

Questão 5 (3,5 pontos):
Para cada algoritmo: Indique, em notação assintótica, o número de comparações necessárias para ordenar uma sequência de n elementos.

	Melhor caso	Pior caso
Selection Sort	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n)$	$O(n^2)$
Merge Sort	$O(n \log n)$	$O(n \log n)$
QuickSort (versão de particionamento sem escolha aleatória do pivô)	$O(n \log n)$	$O(n^2)$

Questão 6 (3 pontos)
Modifique o algoritmo InsertionSort para que, no lugar de ordenar todos os números, apenas sejam ordenados os k primeiros digitos.
O objetivo é obter apenas os k menores valores.
Responda:
- Qual o número de comparações no melhor caso?
- Qual o número de comparações no pior caso?

```
void InsertionSort (int v[], int n) {
    int i, j, aux;
    for (i=1; i<n; i++) {
        aux = v[i];
        for (j=i-1; j>=0 && v[j]>aux ; j--) {
            v[j+1] = v[j];
        }
        v[j+1] = aux;
    }
}
```

```
void InsertionSort (int v[], int n, int k) {
    int i, j, aux, inicio;

    for (i=1; i<n; i++) {
        aux = v[i];

        if (i>k-1)
          inicio = k-1;
        else
          inicio = i-1;

        for (j=inicio; j>=0 && v[j]>aux ; j--) {
            v[j+1] = v[j];
        }
        v[j+1] = aux;
    }
}
```

Número de comparações

Melhor caso:

$$C(n) = n - 1$$

Pior caso:

$$C(n) = kn - \frac{k}{2} - \frac{k^2}{2}$$

Questão 7 (3 pontos)
Qual o cuidado na implementação do algoritmo de ordenação Quick Sort que garante que o espaço utilizado além do vetor a ser ordenado seja logarítmico no tamanho da entrada? Justifique a sua resposta.

A versão tradicional do algoritmo de ordenação QuickSort permite ordenar um vetor de elementos particionando-o em duas partes. O processo de ordenação é realizado de forma recursiva para cada uma das partes.
Dependendo do elemento selecionado como pivô, o particionamento pode ser realizado de forma desbalanceada (isto é, uma das partes pode ter n-1 elementos), assim, no pior dos casos, o número de chamados para o algoritmo Quicksort será linear no tamanho da entrada, sendo necessário espaço linear para registrar os chamados na pilha do computador.
O cuidado que deve ser considerado para manter uma pilha inferior ou igual a log(n) é que, após o particionamento dos elementos, a menor parte (do particionamento) deve ser ordenada.

Questão 8 (3 pontos)
Esboce, em texto corrido, um algoritmo que permita particionar um vetor de inteiros em duas partes (os menores e os maiores). O algoritmo deve devolver o índice do elemento que particiona o vetor. O número de comparações deve ser linear no número de elementos.

Ver aula 7 (métodos eficientes de ordenação)

Questão 9 (3 pontos)
Descreva a estratégia utilizada no algoritmo de ordenação Quick Insertion Sort. Apresente o algoritmo em pseudocódigo ou na linguagem C.

Ver aula 8 (Atividade prática Quick Insertion Sort)