Universidade Federal do ABC
Bacharelado em Ciência da Computação
Algoritmos e Estrutura de Dados I - Prof. Fabrício Olivetti de França e Paulo Henrique Pisani
Turma A1/A2 Diurno - Prova P2 - A

NOME/RA : .....

Instruções: responda as seguintes questões na folha de prova. Entregue as duas folhas com nome e RA.

Questão 01 (2.0 pto). Escreva passo a passo a construção de uma árvore AVL ao inserir a seguinte sequência **nessa ordem**: $3, 7, 10, 12, 8, 9, 1, 11, 6$. Após a construção da árvore, descreva o passo a passo para remover o nó 8.

https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

Questão 02 (3.0 pto). Como você implementaria uma versão do *MergeSort* de tal forma que ele se torne *in-place*. Qual a complexidade dessa nova versão?

```
void merge(registro *base, int m, int n) {
    registro x;
    int j=0, k=m, l;

    for (int i=0; i<n; ++i) {
        // já está ordenado
        if (j==m || k==n) return;

        // aplica a função insert no elemento k
        else if (base[j].key > base[k].key) {
            x = base[j];
            for (l=j; l<k; l++) base[l] = base[l+1];
            base[k] = x;
        }

    }
}
```

Complexidade se torna $O(n^2)$.

Questão 03 (3.0 pto). Altere o algoritmo *QuickSort* para usar como *pivot* a mediana entre k pontos aleatórios, sendo k um parâmetro da função.

```
int mediana(registro *base, int n, int k) {
    registro vals[k];
    int j;
    for (int i=0; i<k; i++) {
        j = rand() / (RAND_MAX /n + 1);
```

```
        vals[i].data = j;
        vals[i].key = base[j].key;
    }
    insertSort(vals, k);
    // temos que pegar um elemento existente,
    // entao pega sempre n/2
    int meio = n/2;

    // retorno o indice correspondente
    return vals[meio].data;
}

int partition(registro *base, int n, int k) {
    int idx = mediana(base, n, k);
    swap(base, base+idx);
    registro pivot = base[0];
    int i=1, j;

    for (j=1; j<n; j++)
    {
        if (base[j].key < pivot.key)
        {
            swap(base+i, base+j);
            ++i;
        }
    }
    swap(base+i-1, base);
    return i-1;
}

void quickSort(registro *base, int n) {
    if (n > 0)
    {
        int p = partition(base, n);
        quickSort(base, p);
        quickSort(base + p + 1, n - p - 1);
    }
}
```

Questão 04 (2.0 pto). Como ficaria o esquema de indexação para uma *Heap* ternária (com 3 filhos)?

$3 * i + 1, 3 * i + 2, 3 * i + 3$