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 - B

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**: $8, 10, 1, 2, 9, 7, 3, 5, 4$. 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). Crie um caso de entrada de dados que represente o pior caso do algoritmo *QuickSort* quando o pivot escolhido for a mediana entre `x[0], x[mid], x[n-1]`, com `mid` sendo o índice do meio da *array* e `n` o tamanho dela.

135624780

0*mid*1*mid*3562478

0*mid*1*mid*2*mid*3*mid*56478

0*mid*1*mid*2*mid*3*mid*4*mid*5*mid*678

0*mid*1*mid*2*mid*3*mid*4*mid*5*mid*6*mid*7*mid*8

Questão 03 (3.0 pto). Altere o algoritmo *MergeSort* para particionar a lista de entrada em 3 partes.

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

    for (int i=0; i<n; ++i) {
        if (j==m)                          x[i] = base[k++];
        else if (k==n)                     x[i] = base[j++];
        else if (base[j].key < base[k].key) x[i] = base[j++];
        else                               x[i] = base[k++];
    }
    for (int i=0; i<n; i++) base[i]=x[i];
    free(x);
}

void mergeSort(registro *base, int n) {
    if (n > 1)
```

```
    {
        int third = n/3;
        mergeSort(base, third);
        mergeSort(base + third, third);
        mergeSort(base + 2*third, n - 2*third);

        /* podemos aplicar o merge duas vezes: 1 para os dois primeiros,
           e outra para o resultado anterior e o último terço */
        merge(base, third, 2*third);
        merge(base, 2*third, n);
    }
}
```

Questão 04 (2.0 pto). Como ficaria a inserção em uma *Heap* ternária (com 3 filhos)? Ilustre com a sequência da *Questão 01*.

- Insere no final da array representando a heap
- Se o elemento for maior que o pai, troca
- Repete recursivamente até a raiz

```
void corrigir(int * heap, int j) {
    int parent = floor((fim-1)/3);
    if(heap[parent] < heap[j]) {
        swap(heap, parent, j);
        corrigir(heap, parent);
    }
}

// n representa o espaco total disponivel
void insertHeap(int x, int * heap, int fim, int n) {
   if (fim==n) return; // nao tem mais espaco
   heap[fim] = x;
   ++fim;
   corrigir(heap, fim-1);
}
```