A atividade corresponde a Avaliação 3 da disciplina de Estrutura de Dados.

Deve ser entregue no formato .DOC ou PDF.

Atividades iguais serão zeradas.

Exercícios que envolvam codificações iguais serão zerados.

Entregas serão aceitas impreterivelmente até as 22h50 do dia 17/06/2021.

Poderão ser feitas arguições a respeito dos exercícios entregues.

1) Dada a estrutura abaixo da classe NO, realize a implementação de um método **recursivo** que adicione um Funcionário em qualquer posição de uma lista duplamente encadeada. Os métodos não devem ter nenhuma iteração, somente chamadas recursivas. A classe Funcionário possui os atributos ID, nome, sobrenome, RG e CPF (3.0 pontos).

```
public class NO {
    public Funcionario dados;
    public NO prox;
    public NO anterior;

    public NO(Funcionario curso){
        dados = curso;
        prox = null;
        anterior = null;
    }
}
```

2) Considerando os algoritmos de ordenação Quick Sort e Merge Sort resolva (1.5 pontos):

a) Explique o funcionamento dos algoritmos Quick Sort e Merge Sort.

b) Simule o algoritmo Quick Sort para o seguinte domínio de entrada: 2, 87, 33, 22, 11, 10, 30, 94, 87, 12, 16, **escolhendo como pivô elemento central.**

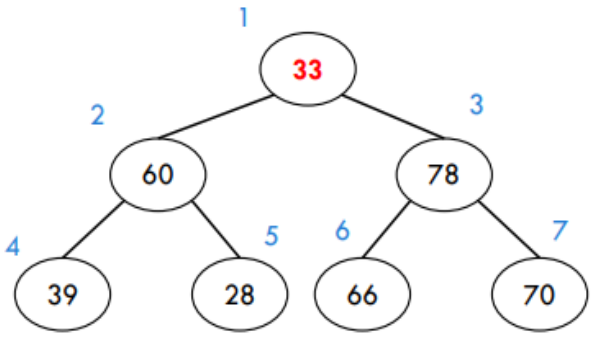
c) Simule o algoritmo Merge Sort para o seguinte domínio de entrada: 3, 5, 6, 7, 91, 78, 33, 42, 55, 17, 9.

d) Explique o funcionamento do método abaixo e qual algoritmo de ordenação ele pertence.

```java
public static void ordenaA (int vet[], int ini, int fim)
{
    int divisao;
    if (ini < fim) {
        divisao = divideElementos(vet, ini, fim);
        ordenaA (vet, ini, divisao-1);
        ordenaA (vet, divisao+1, fim);
    }
}

public static int divideElementos (int vet[], int ini, int fim){
    int p = vet[ini],i = ini+1, f = fim, aux;
    while (i<=f) {
        while (i <= fim && vet[i] <= p)
            ++i;
        while (p < vet[f])
            --f;
        if (i < f){
            aux = vet[i];
            vet[i] = vet[f];
            vet[f] = aux;
            ++i;
            --f;
        }
    }
    if (ini != f){
      vet[ini] = vet[f];
      vet[f] = p;
    }
    return f;
}
```

3) O algoritmo Heap Sort utiliza o conceito de Fila de Prioridades para realizar as operações de inclusão e remoção de elementos. Considerando a ordenação pelo Heap Máximo, **demonstre todos os passos** para a reordenação do algoritmo após a remoção de um elemento (1,5 pontos).



1	2	3	4	5	6	7
33	60	78	39	28	66	70

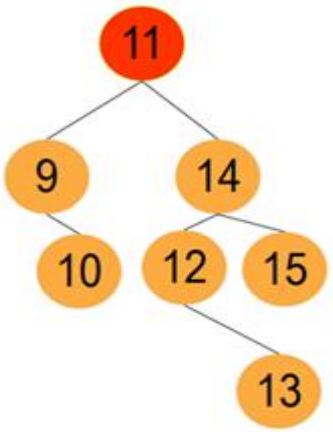
4) Dado os métodos abaixo, explique a qual algoritmo ele pertence. Detalhe o funcionamento e as principais características do algoritmo (1.5 pontos).

```
static void refazOperacao(int A[], int i, int comp) {
    int valE, vaD, m, temp;
    valE = 2 * i + 1;
    vaD = 2 * i + 2;
    if (valE < comp && A[vaD] > A[i]) {
        m = valE;
    } else {
        m = i;
    }
    if (vaD < comp && A[vaD] > A[m]) {
        m = vaD;
    }
    if (m != i) {
        temp = A[i];
        A[i] = A[m];
        A[m] = temp;
        refazOperacao(A, m, comp);
    }
}
static void atualizaValores(int[] A, int i) {
    int n = A.length;
    if (i < n/2){
        atualizaValores(A, 2*i+1);
        atualizaValores(A, 2*i+2);
        refazOperacao(A, i, n);
    }
}
```

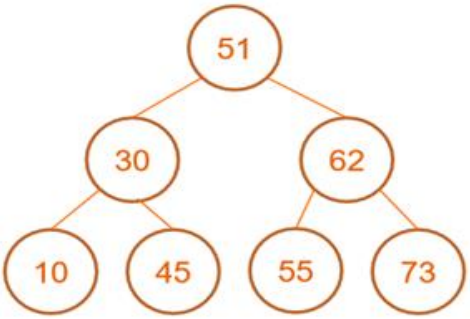
5) Com base no conceito de árvores binárias realize cada um dos exercícios abaixo (1.5 pontos):

a) Simule todos os passos até o estado final da árvore binária para os seguintes elementos: 15, 29, 5, 30, 7, 20, 98, 4. Descreva se a árvore é balanceada ou não balanceada.

b) Dada a árvore binária abaixo, apresente o estado final da árvore ao realizar a remoção do Nó raiz.



c) Apresente os resultados das consultas dos Nós da árvore binária em pré-ordem e pós-ordem, respectivamente.



6) Defina com as suas palavras cada uma das estruturas de dados abaixo e descreva quais são as operações aplicáveis. Simule a remoção de um elemento da posição 2 de uma lista duplamente encadeada que possui os seguintes elementos: 11, 20, 30, 5, 33, 40, 55, 77, 88 (1.0 pontos):

- Ponteiros
- Fila Circular
- Pilha
- Lista Duplamente Encadeada
- Divisão e conquista
- Recursividade direta e indireta
- Busca binaria
- Árvores
- Grafos
- Tabela de Espalhamento