A Atividade será contabilizada como Atividades 5, 6 e 7.

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 23h59 do dia 22/11/2020

1) Defina cada uma das estruturas de dados abaixo, descreva quais são as operações aplicáveis e simule cada uma das estruturas para um conjunto de entrada com 10 elementos (1.5 pontos).

- Lista Ligada
- Pilha
- Fila de Prioridades
- Fila Circular
- Lista Duplamente Ligada
- Árvore Binária
- Grafos

2) Dada a estrutura abaixo da classe NO, realize a implementação de um método **recursivo** que adicione um Curso no final da Lista Duplamente Encadeada e um método **recursivo** que remova e retorne um Curso do final da Lista Duplamente Encadeada. Os métodos não devem ter nenhuma iteração, somente chamadas recursivas. A classe Curso possuí os atributos id, nome, semestre e duração (1.5 pontos).

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

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

3) Considerando o algoritmo Quick Sort simule a sua execução para o seguinte domínio de entrada: [11, 12, 8, 9, 14, 17, 22, 75, 44, 98, 120, 200, 138, 139, 22, 33, 85, 92, 35], **escolhendo como pivô o elemento central** (1.5 pontos).

4) Dado o algoritmo Merge Sort simule a sua execução para o seguinte domínio de entrada: [13, 11, 7, 8, 9, 130, 129, 128, 35, 33, 200, 99, 98, 82, 83, 81, 230, 228] (1.5 pontos).

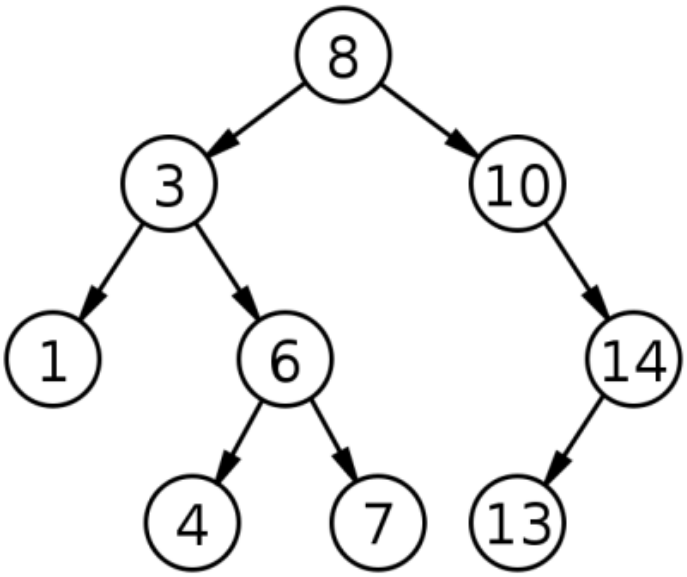
5) Explique o funcionamento dos algoritmos de ordenação Quick Sort, Merge Sort e Heap Sort, detalhe as principais diferenças entre os três algoritmos de ordenação. Explique o funcionamento dos métodos abaixo e qual algoritmo de ordenação eles pertencem (1.5 pontos).

```
public static void ordenaA(int lista[], int inicio,
 int fim){

        if (inicio < fim){
            int meio = (inicio + fim) / 2;
            ordenaA(lista,inicio, meio);
            ordenaA(lista,meio + 1, fim);
            ordenaB(lista,inicio, meio, meio+1,fim);
        }
    }

   public static void ordenaB(int lista[], int inicioA, int fimA,
    int inicioB, int fimB){
           int i1 = inicioA;
           int i2 = inicioB;
           int iaux = inicioA;
           int aux[] = new int[lista.length];
           while (i1 <= fimA && i2 <= fimB){
               if(lista[i1] <= lista[i2])
                   aux[iaux++]=lista[i1++];
               else
                   aux[iaux++]=lista[i2++];
           }
           while (i1 <=fimA)
                   aux[iaux++]=lista[i1++];
           while (i2 <=fimB)
                   aux[iaux++]=lista[i2++];
           for (int i=inicioA;i<=fimB;i++)
               lista[i] = aux[i];
   }
```

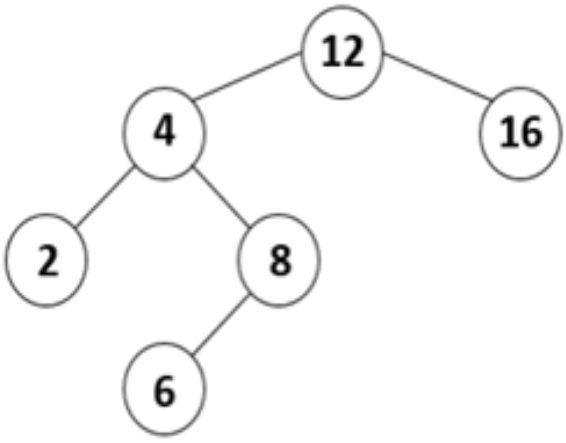
6) Com base no conceito de árvores binárias realize cada um dos exercícios abaixo (2.25 pontos):

a) Simule todos os passos até o estado final da árvore binária para os seguintes elementos: 25, 5, 30, 8, 20, 31, 3, 99, 88, 77, 66.

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.



7) Dadas as afirmações, assinale a alternativa que contém, de cima para baixo, a sequência correta, assinalando V para verdadeiro e F para falso (0.125 pontos). **Justifique a resposta**

( ) A disciplina de acesso da estrutura de dados Pilha determina que o último elemento inserido no conjunto deva ser o primeiro a ser removido.

( ) A implementação de lista utilizando alocação sequencial dos elementos, comparada à alocação encadeada, necessita de mais espaço de armazenamento por elemento do conjunto.
( ) A pesquisa sequencial é mais eficiente que a pesquisa binária para busca de elementos em listas ordenadas implementadas com alocação sequencial dos elementos.
( ) As estruturas de dados Pilha e Fila podem ser implementadas utilizando tanto abordagens baseadas na alocação sequencial quanto na alocação encadeada dos elementos.
( ) A inserção de um elemento no início de uma lista duplamente encadeada implica no deslocamento dos elementos já existentes na memória.
a) V, V, V, F, F.
b) V, F, V, F, F.
c) V, F, F, V, F.
d) F, V, F, V, V.
e) F, F, V, F, V.

8) Existem várias estruturas que podem ser implementadas no processo de desenvolvimento de software. Listas Simplesmente Encadeadas, Listas Duplamente Encadeadas, Filas e Pilhas podem trabalhar com alocações estáticas e dinâmicas de memória. Sobre as Listas Simplesmente Encadeadas é correto afirmar que (0.125 pontos):
a) Sempre nas Listas Simplesmente Encadeadas o último elemento irá possuir um ponteiro para o primeiro elemento da lista, facilitando o acesso e remoção de elementos por meio de um percurso simplificado.
b) Nas estruturas de dados de alocação dinâmica de memória é necessário definir o tamanho no momento de sua criação.
c) Quando um novo elemento for inserido na lista é necessário que os ponteiros dos elementos envolvidos sejam atualizados, sem a necessidade de qualquer operação de deslocamento físico dos demais elementos da lista.
d) Listas encadeadas possuem como principio de funcionamento a atualização dos ponteiros envolvidos, porém cada vez que um novo NO é inserido na lista o endereço de memória de todos os demais elementos precisa ser atualizado para que não haja perda de referências de memória.

e) Na recuperação de um elemento da lista não é necessário percorrer os demais elementos. O acesso ocorre sempre pela posição do elemento na lista.
f) Listas estáticas e listas de alocação dinâmica de memória sempre utilizam ponteiros. Os índices do vetor representam endereços de memória.
g) Quando fazemos a movimentação dos elementos em um vetor, estamos movimentando os endereços de memória e não índices.
h) Todas as anteriores.