1) Em termos de estruturas de dados, defina ocultamento e encapsulamento:

2) Porque ao comparar dois algoritmos para verificar qual é o melhor, não é correto utilizar o tempo de execução como medida de desempenho? Porque o correto a se fazer é medir o número de operações que os algoritmos fazem?

3) No código abaixo a quantidade de operações que um algoritmo executará está em função da entrada. Nesse exemplo, quando ocorre o melhor caso? Qual o pior caso?

```
int BuscaSequencial(int* vetor, int tam, int chave)
{
    int i;
    for(i = 0; i < tam; i++)
    {
        if(vetor[i] == chave)
        {
            return i; //retornar índice ou ponteiro
        }
    }
    return -1;//não encontrou indice;
}
```

4) Explique em palavras o significado de cada uma das notações de complexidade de algoritmos: Big O, Big Ômega, Big Theta.

5) Quando se trata de uma estrutura para armazenar, buscar e remover elementos, quando compensa ordenar uma estrutura para fazer uma busca binária?

6) Defina o conceito de estabilidade em algoritmos de ordenação e de um exemplo prático:

7) Defina o conceito de adaptabilidade de algoritmos de ordenação:

8) Utilizando o Algoritmo BubbleSort Ordene em ordem crescente o seguinte vetor, marcando os elementos que fazem parte da troca:

4	1	5	2	3
1	2	3	4	5

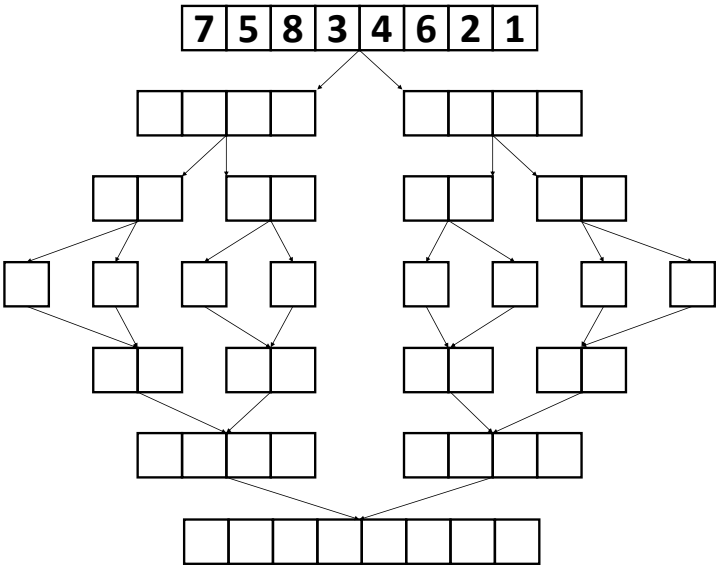
9) Utilizando o Algoritmo InsertionSort Ordene em ordem crescente o seguinte vetor, marcando o elemento selecionado para a inserção:

5	4	8	2	6	1	7	3
1	2	3	4	5	6	7	8

10) Utilizando o Algoritmo SelectionSort Ordene em ordem crescente o seguinte vetor, marcando os elementos que fazem parte da troca:

8	7	6	5	4	3	2	1
1	2	3	4	5	6	7	8

11) Utilizando o Algoritmo MergeSort Ordene em ordem crescente o seguinte vetor:



12) Explique o princípio de geral de funcionamento do algoritmo QuickSort:

13) Explique o princípio de geral de funcionamento do algoritmo MergeSort:

14) Explique o princípio de geral de funcionamento do algoritmo HeapSort:

15) Marque V para verdadeiro e F para falso.

( ) BubbleSort é estável.
( ) InsertionSort é estável.
( ) SelectionSort é estável.
( ) QuickSort é estável.
( ) MergeSort é estável.
( ) HeapSort é estável.

16) Imprima a saída da busca binária a seguir:

```
int v[] = {1,2,3,4,5,6,7,8,9,10};
int N = 10, chave = 3;
int inicio = 0, meio, fim = N-1;
while(inicio <= fim)
{
    meio = (inicio + fim) / 2;
    printf("%d %d %d\n", inicio, meio, fim);
    if(v[meio] < chave) inicio = meio + 1;
    else if(v[meio] > chave) fim = meio - 1;
    else fim = -1;
}
printf("indice: %d\n", meio);
```

17) No código acima, qual ou quais chaves são o pior caso? Qual é a complexidade assintótica da busca binária no pior caso?

18) Sobre Tabela Hash, defina o conceito geral:

19) Sobre Tabela Hash, quando ocorrem colisões, podemos realizar correção por encadeamento, explique:

20) Sobre Tabela Hash, quando ocorrem colisões, podemos realizar correção por endereçamento aberto, cite 3 formas de endereçamento aberto:

21) Considerando que você já tenha o MergeSort implementado, conforme os protótipos abaixo, crie uma struct para receber nome de um arquivo, tipo e data de modificação. Ordene os dados para ficar conforme os exemplos abaixo, faça o código da main e a função de callback. A primeira linha de entrada é um inteiro N representando o número de linhas que vem a seguir. Os dados são informados da seguinte forma: nome, tipo, data. O nome e o tipo deve ser ordenado em ordem nominal, e a data em ordem decrescente.

```
int compara_nome(Bau A, Bau B);
int compara_tipo(Bau A, Bau B);
int compara_data(Bau A, Bau B);
void Merge (Bau vet[],int inicio,int meio,int fim, int (*meu_decisor)(Bau A, Bau B));
void MergeSort (Bau vet[],int inicio,int fim, int (*meu_decisor)(Bau A, Bau B));
```

Exemplo de Entrada	**Exemplo de Saída**
11 testes .txt 10/11/2015 testes .c 11/12/2015 lixo .bat 16/08/2018 arvores .jpg 14/09/2018 flores .jpg 14/09/2018 rosto .jpg 13/09/2018 prova .pdf 14/09/2018 lista .pdf 14/09/2018 Artigo .tex 13/09/2018 Artigo .pdf 13/09/2018 Exemplo .pdf 13/09/2018	arvores .jpg 14/09/2018 flores .jpg 14/09/2018 lista .pdf 14/09/2018 prova .pdf 14/09/2018 rosto .jpg 13/09/2018 Artigo .pdf 13/09/2018 Exemplo .pdf 13/09/2018 Artigo .tex 13/09/2018 lixo .bat 16/08/2018 testes .c 11/12/2015 testes .txt 10/11/2015

22) Faça um programa que leia nome e RA de alunos, armazene-os em structs, ordene os dados de acordo com o RA, e em seguida busque os elementos pelo RA com busca binária. Pode-se fazer isso com qsort e bsearch.