Lab_01

//----------

Prazo de entrega: 24/03/2025

Valor: 0

Não entrega: -5 valores

//----------

1. Considere a função abaixo:

```
1. int soma_vetor(int arr[], int n) {
2.     int soma = 0;
3.     for (int i = 0; i < n; i++) {
4.         soma += arr[i];
5.     }
6.     return soma;
7. }
```

Identifique em cada linha de instrução a frequência com que esta ocorre, preenchendo a tabela abaixo. Calcule a complexidade no pior caso, justificando.

Instrução	**Subtotal**
`int soma_vetor(int arr[], int n) {`	
`    int soma = 0;`	
`    for (int i = 0; i < n; i++) {`	
`        soma += arr[i];`	
`    }`	
`    return soma;`	
`}`	
Total	
Complexidade no pior caso	

2. Considere a função abaixo:

```
1. void soma_matrizes(int rows, int cols, int a[rows][cols],
                      int b[rows][cols], int c[rows][cols]) {
2.     for (int i = 0; i < rows; i++) {
3.         for (int j = 0; j < rows; j++) {
4.             c[i][j] = a[i][j] + b[i][j];
5.         }
6.     }
7. }
```

Identifique em cada linha de instrução a frequência com que esta ocorre, preenchendo a tabela abaixo. Calcule a complexidade no pior caso, justificando.

Instrução	Subtotal

`void soma_matrizes(int rows, int cols,`
`int a[rows][cols],`
`int b[rows][cols],`
`int c[rows][cols]) {`	

`for (int i = 0; i < rows; i++) {`	
`for (int j = 0; j < cols; j++) {`	
`c[i][j] = a[i][j] + b[i][j];`	
`}`	
`}`	
`}`	
Total	
Complexidade no pior caso	

3. Implemente os algoritmos de procura sequencial e binária utilizando funções, cujas declarações/assinaturas devem ser:
   int procura_sequencial(int arr[], int n, int elem);
   int procura_binaria(int arr[], int n, int elem);

   a. Obtenha os tempos de execução para 5 valores diferentes de n, para cada um dos dois algoritmos. Apresente os tempos e os valores de n numa tabela.
   b. Corra o programa nos diferentes portáteis dos 4 elementos do grupo. Apresente os tempos e os valores de n, complementando a tabela já criada.
   c. Gere um gráfico n X tempo de execução para cada uma das duas funções. Identifique com cores diferentes os pontos relativos à simulação de cada elemento do grupo.
   d. Proponha ou pesquise um outro algoritmo de procura e repitas os passos anteriores (implemente, calcule os tempos de execução nos diferentes computadores dos elementos do grupo e inclua no gráfico anterior).
   e. Faça o estudo da complexidade no pior caso dos três algoritmos.
   f. Escreva as conclusões do grupo sobre esta experiência (O que aprenderam com isto? Qual foi o objetivo do estudo?).

Utilize alocação dinâmica. Peça ao utilizador para digitar o tamanho do array (n) e então aloque dinamicamente o array com n elementos na função main().

Lembre-se que no caso da procura binária o array deve estar ordenado.

Para obter os tempos do experimento, utilize os comandos clock() e difftime(tempo2, tempo1) da biblioteca time.h.

Para obter o tempo em segundos, calcule clock()/CLOCKS_PER_SEC, sendo:

- clock: tempo do processador em clocks;

- CLOCKS_PER_SEC: constante que fornece o número de clocks em um segundo;
- O tempo em clocks é de um tipo de dados especial, chamado clock_t.
- Exemplo:

```
...
int main(){
	...
	clock_t tempo1, tempo2;
	double tempo_total;
	...
	tempo1 = clock();
	//sua implementação
	tempo2 = clock();

	tempo_total = difftime(tempo2,tempo1)/ CLOCKS_PER_SEC;
	printf("Tempo da procura XX: %g:\n", tempo_total);
	...
}
```