O programa deverá ser escrito na linguagem C e deve ser compilado e executado

A ideia é implementar um programa para comparar os diferentes algoritmos de ordenação que foram discutidos durante a disciplina. Portanto, deverão ser implementados os seguintes métodos de ordenação:

1. Bolha
2. Bolha melhorado
3. Seleção
4. Inserção
5. Merge Recursivo
6. Merge Iterativo
7. Quick Sort
8. Bucket Sort
9. Contagem (apenas para números aleatórios)

Ao iniciar o programa o usuário deve ser capaz de escolher:

1 - Tamanho do conjunto de dados (vetor) que será ordenado, podendo variar entre 1 e 100.000

2 - Definir o conjunto de dados que será ordenado, podendo ser:

1. Aleatório (Qualquer valor entre 0 e 999)
    a. Exemplo: 23 45 999 654 389 879 9 5 45 22 87 13 11 110 442 555

2. Crescente
    a. Exemplo: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

3. Decrescente
    a. Exemplo: 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

4. Quase ordenado (Inserir um valor aleatório a cada 10 números)
    a. Exemplo: 0 1 2 3 4 5 6 7 8 9 492 11 12 13 14 15 16 17 18 19 390

Após a definição do tamanho do conjunto e de como esse conjunto estará disperso, deverão ser executados todos os algoritmos de ordenação, um após o outro, escrevendo o tempo de execução de cada um na tela conforme exemplo abaixo:

Exemplo de execução quando o usuário escolher 100.000 números aleatórios

```
Digite o tamanho do vetor que deve ser ordenado [1-100000]: 100000
Digite o tipo de conjunto de dados a ser ordenado:
1 - Aleatorio
2 - Ordenado Crescentemente
3 - Ordenado Decrescentemente
4 - Quase Ordenado
Opcao: 1
Inicializando os vetores!

Comparando ordenação de valores aleatórios
Metodo Bolha: 21.187155s
Metodo Bolha Melhorado: 21.883271s
Metodo Insertion Sort: 4.773282s
Metodo Selection Sort: 8.367866s
Metodo Merge Sort: 0.013240s
Metodo Merge Sort Iterativo: 0.012719s
Metodo Quick Sort: 0.008813s
Metodo Bucket Sort: 0.039714s
Metodo Contagem: 0.000551s
```

Obs: Esse foi o tempo que de execução em uma máquina específica, diferentes máquinas terão tempos de execução diferentes, porém respeitando a mesma ordem de grandeza na diferença de execução entre os diferentes algoritmos.

A avaliação utilizará o seguinte quadro de notas

0 Erro de compilação e/ou não rodar com nenhum algoritmo e nenhum conjunto de dados e/ou não mostrar o tempo de execução dos algoritmos. Ou seja, só receberá nota diferente de zero caso apresente ao menos algum resultado (em segundos) do tempo que levou para realizar uma ordenação correta conforme solicitado pelo usuário
1 Ter 9 falhas entre os algoritmos/tamanho/conjunto de dados
2 Ter 8 falhas entre os algoritmos/tamanho/conjunto de dados
3 Ter 7 falhas entre os algoritmos/tamanho/conjunto de dados
4 Ter 6 falhas entre os algoritmos/tamanho/conjunto de dados
5 Ter 5 falhas entre os algoritmos/tamanho/conjunto de dados
6 Ter 4 falhas entre os algoritmos/tamanho/conjunto de dados
7 Ter 3 falhas entre os algoritmos/tamanho/conjunto de dados
8 Ter 2 falhas entre os algoritmos/tamanho/conjunto de dados
9 Ter apenas uma falha entre os algoritmos/tamanho/conjunto de dados
10 Rodar perfeitamente sem nenhum bug com todos os conjuntos de dados, todos algoritmos de ordenação e com tamanho de 1 a 100.000

Também será descontado 1 ponto para:
Cada warning durante a compilação;
Se os vetores a serem ordenados não forem idênticos:
o Exemplo: O usuário escolheu o tamanho 10 e aleatório então um algoritmo de ordenação ordena o vetor 75 5 8 7 3 99 845 124 946 451 e outro algoritmo ordena o vetor 85 15 74 67 58 49 243 52 6 7