INSTITUTO FED. DE EDUCAÇÃO, CIÊNC. E TEC. DE PERNAMBUCO
CURSO: TEC. EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: ALGORITMOS E ESTRUTURAS DE DADOS
PROFESSOR: RAMIDE DANTAS
ASSUNTO: ORDENAÇÃO

Aluno (a):			
Matrícula:		**Data:**	

Prática 07

Parte 0: Preparação

Passo 1: Crie um novo projeto chamado **Pratica7**.

Configure o projeto para utilizar C++11.
(Project Properties > C++ Build > Settings: no caso do Cygwin procure por "Dialect" e selecione "ISO C++11" em "Language standard")

Passo 2: Utilize o arquivo **ordenacao.cpp** que acompanha a prática 7 ao projeto.

Esse arquivo é uma versão levemente simplificada da apresentada em sala de aula.

Parte 1: Implementando BubbleSort

Passo 1: Implemente a função `bubblesort()` do arquivo **ordenacao.cpp**.

Siga o algoritmo contido no material de aula.

Passo 2: Rode e teste a aplicação.

Verifique se a ordenação ocorreu adequadamente. Use a função `print()` e o depurador para identificar erros na sua implementação. Modifique o número de elementos do array para ver o efeito no tempo de execução.

Parte 2: Implementando InsertionSort / SelectionSort

Passo 1: Escolha entre implementar a função `selectionsort()` ou `insertionsort()`.

Siga a descrição dos algoritmos contido no material de aula.

Passo 2: Rode e teste a aplicação.

Verifique o resultado e faça a depuração se necessário.

Parte 4: Implementando a junção do MergeSort

Passo 1: Implemente a função `merge()`, usada pela função `mergesort()`.

Essa função realiza a junção dos dois subarrays ordenados de `buffer` em `target`. A primeiro pedaço vai de `start` até `mid - 1` e o segundo de `mid` até `finish`. O array `target` é indexado de `start` até `finish` (inclusivo).

Passo 2: Rode e teste a aplicação.

Verifique o resultado e faça a depuração se necessário.

Parte 4: Implementando a partição do QuickSort

Passo 1: Implemente a função `partition()`, usada pela função `quicksort()`.

> Essa função deve organizar o array de forma que o elemento escolhido como pivô acabe na sua posição correta (ordenada), todos os elementos a sua esquerda são menores (ou iguais) a ele, e os da direita são maiores. A função por fim retorna a posição final do pivô. Tome cuidado para não violar os limites do array (`start` e `finish`). Siga a descrição da função no material de aula se necessário.

Passo 2: Rode e teste a aplicação.

> Verifique o resultado e faça a depuração se necessário.

Passo 3: (Desafio) Implemente versões alternativas da função `partition()` usando diferentes formas de selecionar o pivô e avalie o impacto disso no desempenho do algoritmo.

> Há variações da função de partição onde o pivô não necessariamente termina na sua posição correta. Isso requer modificar a função principal `quicksort()` de acordo.