Segundo Trabalho Computacional – Valor 1,5

Descrição do Trabalho:

1) Parte I:

• O trabalho tem como objetivo comparar o **tempo de processamento** de seis algoritmos de ordenação (**Bubble Sort**, **Insertion Sort**, **Selection Sort**, **Quicksort**, **Merge Sort** e **Heap Sort**) aplicados a quatro vetores **A**, **B**, **C** e **D** com, respectivamente, 5.000, 10.000, 20.000 e 30.000 números inteiros. Os vetores têm que estar armazenados nos arquivos "**A.txt"**, "**B.txt"**, "**C.txt"** e "**D.txt"**, respectivamente.

• Os vetores têm que ser gerados aleatoriamente com números de 1 até 10.000.

• Todos os vetores utilizados têm que ser declarados na função *main*.

• O grupo deverá utilizar a **implementação** (**na linguagem C**) **estudada em sala** para os quatro primeiros algoritmos (**Bubble Sort**, **Insertion Sort**, **Selection Sort** e **Quicksort**) e pesquisar implementações (**na linguagem C**) dos demais (**Merge Sort** e **Heap Sort**).

• Cada algoritmo deve ordenar cada vetor de maneira **NÃO-CRESCENTE**.

• Três **procedimentos**, no mínimo, têm que ser utilizados no código de cada algoritmo:

1) ler o vetor do respectivo arquivo;

2) ordenar o vetor de maneira ***não-crescente***;

3) imprimir em um **arquivo** o vetor ordenado. Utilize um arquivo de saída para cada vetor: "**Ordenado-A.txt"**, "**Ordenado-B.txt"**,"**Ordenado-C.txt"** e "**Ordenado-D.txt"**.

• Para calcular o **tempo de processamento** (em **segundos**), pesquise um procedimento ou função para esta finalidade. Inicie o cálculo do tempo de processamento antes do procedimento de **ordenar** o vetor e finalize o cálculo após o procedimento de **ordenar** o vetor:

```
lerVetor( );
inicioCalculaTempo( );
ordenaVetor( );
fimCalculaTempo( );
imprimeVetorOrdenado( );
```

• Para análise dos resultados, uma tabela tem que ser preenchida com os tempos de processamento (em **segundos**) gasto por cada algoritmo para ordenar cada vetor. Um exemplo de tabela pode ser visto a seguir:

	Bubble	**Insertion**	**Selection**	**Quicksort**	**Merge**	**Heap**
Vetor A						
Vetor B						
Vetor C						
Vetor D						

2) Parte II:

• Explicar, **passo a passo**, como os algoritmos **Mergesort** e **Heapsort** ordenam o vetor a seguir de maneira **NÃO-CRESCENTE**:

5	1	9	0	8	3	1	4	6	2

O que deve ser enviado ao Professor:

• O código de cada algoritmo (Bubblesort.c, Insertionsort.c, Selectionsort.c, Quicksort.c, Mergesort.c e Heapsort.c);

• Os arquivos dos quatro vetores (“A.txt”, “B.txt”, “C.txt” e “D.txt”);

• Um arquivo (.pdf) contendo:

- **especificação da máquina utilizada nos testes**: se é desktop ou notebook, processador e clock, sistema operacional e a versão, arquitetura (64 ou 32 bits) e a quantidade de memória RAM;

- **apresentação dos resultados**: a tabela descrita anteriormente e um texto informando qual algoritmo apresentou melhor desempenho (em relação ao tempo de processamento) para cada vetor;

- a explicação **passo a passo** da execução dos algoritmos **Mergesort** e **Heapsort** sobre o vetor apresentado.

OBS.: Os arquivos (as seis implementações (.c), os quatro vetores (.txt) e o arquivo .pdf) têm que ser enviados **compactados**. Para isso, coloque-os em uma pasta e em seguida faça a compactação da mesma.

Importante:

- O grupo tem que ser o mesmo utilizado para o T1.

- O grupo deve entregar o trabalho (**Valor 1,0**) até às **23h59** do dia **12/12/2021** (domingo) via e-mail para:

philippeleal@yahoo.com.br

- O grupo realizará a apresentação oral do trabalho (**Valor 0,5**) na aula síncrona do dia **16/12/2021**, quando **todos os componentes do grupo devem estar presentes** e o Professor poderá realizar perguntas sobre o trabalho para qualquer componente.

- Caso algum componente do grupo não esteja presente na aula apresentação do trabalho, o mesmo ficará **sem o ponto do trabalho (Valor 1,5)**.

- Apenas um e-mail por grupo deve ser enviado. Isto é, não há a necessidade de cada componente do grupo enviar. Porém, **é recomendável enviar uma cópia do e-mail para os outros componentes**.

- Algoritmos que não estão compilando não serão aceitos. Portanto, não é necessário enviar, receberão nota **ZERO**. Caso o grupo esteja com dúvidas, tire-as com o professor **EM SALA (NÃO POR E-MAIL)** ou com os monitores. Não deixe para a última hora.

- O **ASSUNTO** do e-mail deve ter a seguinte formatação:

PE-Quinta-T2-PrimeiroNomedosAlunos

Exemplo de e-mail do grupo (fictício) formado pelos alunos Lucas Pereira, Renato da Silva e Miguel dos Santos:

Para: philippeleal@yahoo.com.br

De: Lucas Pereira

Assunto: PE-Quinta-T2-Lucas-Renato-Miguel

Anexo: PE-Quinta-T2-Lucas-Renato-Miguel.zip

Repare que o nome do arquivo compactado (.zip, .rar) deve ter a mesma formatação do "Assunto" do e-mail:
PE-Quinta-T2-Lucas-Renato-Miguel.zip

- A primeira linha de cada código deve conter os nomes dos **componentes do grupo**.

- Utilize nomes sugestivos para as variáveis. Faça corretamente a identação e comentários no código para facilitar seu entendimento. Estes itens serão avaliados.

- A chave de fechamento de um comando deve estar na mesma direção do comando que fez a abertura. Por exemplo:

```
if(x > 0){


} //O fechamento deve estar alinhado ao comando!!
```

- Preste atenção se está enviando a versão correta do trabalho, **visto que será considerada a data da entrega da versão correta**.

- Não deixe para enviar o trabalho na última hora, pois podem acontecer problemas com o envio.

- **Os trabalhos serão testados no Mint 19.3 com gcc versão 7.5.0**.

- Preste atenção na hora (**23h59**) e na data de entrega (**12/12/2021**), **pois não serão aceitos trabalhos recebidos após o prazo estipulado**.