INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PARAÍBA Campus Campina Grande	**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA – CAMPUS CAMPINA GRANDE**		
	CURSO:	**CURSO SUPERIOR ENGENHARIA DA COMPUTAÇÃO**	
	PERÍODO:		**TURMA: A,B**
	DISCIPLINA:	**ESTRUTURAS DE DADOS**	
	PROFESSOR:	**CÉSAR ROCHA VASCONCELOS**	SEMESTRE LETIVO

Descrição do Projeto Final (versão 1.0)

1. **Informações gerais do projeto:**

   Usando a linguagem C, **o trio** (*no máximo, três alunos! Se, no dia da entrega, houver uma equipe com mais de três alunos, o projeto não valerá mais nada!*) deve apresentar um projeto prático, conforme instruções a seguir. Usando a linguagem C, você deve **implementar uma biblioteca contendo um algoritmo de ordenação** (vide temas a seguir). Este pequeno projeto está previsto para ser construído até término do período, com temas e datas de entrega detalhadas.

2. **Descrição geral dos temas de trabalho e outras informações:**

   A equipe deverá construir uma biblioteca única e diferente[1] dos demais colegas. Cada tema será atribuído a uma equipe por meio de um sorteio simples. No geral, conforme dito anteriormente, cada biblioteca irá modelar algum algoritmo de ordenação relacionado à disciplina de Estrutura de Dados. Este projeto também possui extras, que sua equipe pode, opcionalmente, fazer para turbinar a nota final.

   Existem algoritmos clássicos da Computação que têm como objetivo ordenar de maneira eficiente um conjunto de dados quaisquer (isto em vetores, listas etc.). Cada equipe que, por sorteio, receber um algoritmo, deverá implementar uma biblioteca (com boa documentação do código) contendo **a implementação** (mais uma vez: comente bastante o arquivo .h e a implementação), vantagens, desvantagens e o uso deste algoritmo. Este projeto deve não somente implementar o algoritmo de ordenação, mas incluí-lo em uma biblioteca para potencializar o reuso. A partir daí, deve ser criado um arquivo *main* que irá importar a biblioteca de ordenação e demonstrar o uso deste algoritmo de ordenação. Detalhe: o *main* deve, primeiro, gerar um conjunto de números inteiros a esmo (vide API de linguagem C para encontrar algo que possa lhe ajudar a gerar estes números randomicamente). O tamanho deste conjunto de números aleatórios deve ser igual a 3 milhões de elementos. Este conjunto de dados pode conter repetições e deve ser armazenado tanto em um vetor simples como uma lista encadeada. Ademais, o programa também deve "exibir" na tela: (a) o conjunto de elementos desordenado, (b) o conjunto de elementos após a execução do algoritmo e (c) quanto tempo o algoritmo levou para ordenar este conjunto de 3 milhões de números (vide API de linguagem C para encontrar algo que possa lhe ajudar a marcar o tempo antes e depois da ordenação terminar e exibir o período de tempo decorrido). Dica: implemente variações de sub-rotinas de ordenação de forma que recebam: *arrays* e listas encadeadas como parâmetros.

   Os seguintes temas de trabalho deverão ser abordados:

   TEMA(1). **Algoritmo Bucket Sort** (31/05) Turmas B e A

   TEMA(2). **Algoritmo Buble Sort** (31/05) Turmas B e A

   TEMA(3). **Algoritmo Selection Sort** (03/06 Turma B) , (04/06 Turma A)

   TEMA(4). **Algoritmo Insertion Sort** (03/06 Turma B) , (04/06 Turma A)

[1] A redundância, aqui, é necessária e serve como um alerta para o caso de cópias entre projetos nas turmas A e B.

TEMA(5). **Algoritmo Merge Sort** (07/06) Turmas B e A

TEMA(6). **Algoritmo Quick Sort** (07/06) Turmas B e A

TEMA(7). **Algoritmo Heap Sort** (10/06 Turma B) , (11/06 Turma A)

TEMA(8). **Algoritmo Counting Sort** (10/06 Turma B) , (11/06 Turma A)

TEMA(9). **Algoritmo Radix Sort** (14/06) Turmas B e A

TEMA(10). **Algoritmo Shell Sort** (14/06) Turmas B e A

3. Cronograma:

10/05: Apresentação dos projetos.
13 à 17/05: Listas de exercícios de recursividade e árvores.
20 e 21/05: Tempo livre para implementar seus projetos.
24/05: 3ª Prova (Árvores & Recursividade)

17/06: Reposição só para quem faltou alguma prova (Turma B)
18/06: Reposição só para quem faltou alguma prova (Turma A)
28/06: Prova Final

4. Sobre a avaliação:

Cada projeto valerá **até** (veja que eu escrevi "até") 100 pontos e irá ser computado em apenas 01 dentre as quatro avaliações (no caso, sua última nota) da disciplina. Cada projeto deverá ser entregue ao professor na data do seu tema, sem atrasos, ou não valerá mais nada. Vale lembrar que o projeto não é opcional, ou seja, se você resolver não apresentar o projeto, ficará com ZERO nesta nota (e não terá nenhuma nota baixa anulada)! Não haverá uma nova data para entrega de projetos. Além disso, não há reposição de projetos.
Cada defesa não deve conter slides e/ou relatórios, mas apenas o programa e eventuais artefatos devem ser mostrados com clareza para o professor e a turma. Além disso, o projeto deve ser defendido pela equipe, isto é, todos os integrantes devem explicar alguma parte do projeto igualmente distribuída. Seu grupo terá, no máximo, 30min para apresentar sua solução de *software* para a turma! Na apresentação, a nota de cada aluno não será dada pelo desempenho da equipe como um todo, mas pelo desempenho individual de cada aluno na hora da apresentação. Assim, se um integrante da equipe não explicar nada (ou quase nada), ou não explicar direito sua parte durante a apresentação, ficará com uma nota baixa.

O temas de projetos serão avaliados em duas grandes frentes:

Implementação: (a) usando comentários na biblioteca, o aluno conseguiu explicar (com profundidade) todas as linhas de código do algoritmo de ordenação na biblioteca? (b) utilizou conceitos de modularização e reuso com bibliotecas (o *main* consegue facilmente importar sua biblioteca e utilizá-la normalmente?); (c) o algoritmo funcionou corretamente? (conseguiu ordenar com sucesso o conjunto de dados, seja com vetor ou com lista encadeada?); (d) foi mostrado o tempo que o algoritmo levou para ordenar tudo?. Qual a conclusão?

Apresentação: (a) o grupo começou sua defesa na hora combinada ou atrasou bastante para começar? (b) todos os integrantes participaram? (c) o grupo conseguiu apresentar o projeto em no máximo 30min? (d) o programa apresentou algum tipo de problema durante a apresentação (travamentos, situações inesperadas, etc.)?

Perguntas do professor: (a) o grupo conseguiu responder às questões levantadas pelo professor, com clareza?

5. O que seu grupo deve entregar ao final da apresentação?

- No dia da apresentação do seu projeto, sua equipe deve disponibilizar um repositório público no *GitHub* com todo o código-fonte da sua aplicação e eventuais artefatos relacionados aos opcionais (caso sua equipe tenha feito algum). O nome da equipe e o link do repositório com o código final deve ser comunicados ao professor no dia da apresentação.
- Se sua equipe não dispõe de um *notebook* para executar seu programa, grave um vídeo da execução do algoritmo.

IMPORTANTE: não lançarei notas em projetos não apresentados, não entregues, tentativas de entrega apenas por e-mail sem defesa ou até mesmo por um "colega-aluno-procurador". Além disso, não aceitarei desculpas como, por exemplo: a) "eu trabalho o dia todo e não tive tempo de fazer o projeto; b) na minha máquina de casa, estava rodando tudo certo, mas, agora, não sei a razão do sistema estar agindo assim; c) tive de viajar a trabalho no dia da defesa do projeto; d) minha avó ficou muito doente no dia da defesa do projeto e eu não consegui entregar; d) o meu hd pegou fogo com todo o meu projeto dentro; ou e) meu cachorro comeu meu *pendrive* com todos os meus arquivos dentro; etc. etc. etc.", tá?

Além disso, não é permitido sair da sua equipe e ingressar em outra em hipótese alguma; mesmo no caso de desistências. Se houver ingresso de integrantes formando uma nova equipe, TODOS os integrantes ficarão sem nota (mesmo a equipe que "acolheu o integrante desgarrado"). Também não é permitido, após o sorteio, querer apresentar seu projeto usando dia e horário da outra turma de estrutura de dados em hipótese alguma! Está avisado!

6. Sobre os extras:

6.1 A fonte de dados desordenada lida a partir de um arquivo texto de entrada, bem como o conjunto ordenado de dados final escrito em um arquivo texto de saída. Lembrar que as funções de gravação e leitura de dados não devem estar engessadas à biblioteca de ordenação (+2,5 pontos)
6.2 Uma interface gráfica completa (com botões, etc.) para o programa de ordenação: (+4,0 pontos)
6.3 Explicar qual é a complexidade do algoritmo para a turma (+ 1,5 pontos)
6.4 Usar o tipo `bool` e a biblioteca `<stdbool.h> (c99)` para representar os valores booleanos da sua aplicação, em vez dos tradicionais 0´s e 1´s (+1,5 pontos)