Tarefa 1 – Geometria Computacional

Aluno: **Matrícula:**

1) Objetivos:

O objetivo dessa tarefa é fazer exercícios e algumas implementações sobre o modelo de complexidade computacional aplicado à geometria computacional (Unidade 1 da cadeira).

2) Organização:

Essa tarefa é obrigatória e cada aluno deve fazê-la individualmente. A tarefa deve ser colocada em local definido pelo professor. Os códigos devem ser feitos em C++ e Linux. Pode-se opcionalmente fazer essa tarefa usando Java e Windows, ou em uma outra linguagem e ambiente acertados com o professor. Os exercícios devem ser feitos em editor de textos (tipo WORD ou outro) ou então em papel e escaneados. Depois deve ser gerado um PDF que deve conter as questões resolvidas, junto com os códigos desenvolvidos. Os códigos devem também ser entregues, assim como os executáveis. Executáveis devem incluir todas as bibliotecas usadas. Todos os arquivos, incluindo fontes, executáveis e os exercícios, devem estar juntos em um único arquivo compactado, a ser entregue pelo aluno.

3) O que entregar:

Um único arquivo compactado contendo:
a) Um PDF com todos os exercícios resolvidos.
b) Código fonte das implementações desenvolvidas.
c) Executável das implementações desenvolvidas.

OBS: Recomenda-se que o executável não tenha nada dinâmico, ou seja, que as LIBs sejam estáticas ou todas as DLLs estejam incluídas na distribuição do arquivo.

4) Quando entregar:

No dia e local a ser definido pelo professor da disciplina.
Qualquer atraso na entrega da tarefa não será permitido.

OBS: Não enviar nenhuma tarefa para email do professor!

5) Questões:

Questão 1:
Implemente o algoritmos de ordenação por seleção, testando com um exemplo com 10, 100 e 1000 elementos gerados de forma aleatória. Além disso, teste com um exemplo já ordenado de 10 elementos. Faça uma análise dos resultados obtidos na sua implementação.

Questão 2:
Implemente o algoritmos de ordenação por quicksort, testando com um exemplo com 10, 100 e 1000 elementos gerados de forma aleatória. Além disso, teste com um exemplo já ordenado de 10 elementos. Faça uma análise dos resultados obtidos na sua implementação.

Questão 3:
Implemente o algoritmos de ordenação por mergesort, testando com um exemplo com 10, 100 e 1000 elementos gerados de forma aleatória. Além disso, teste com um exemplo já ordenado de 10 elementos. Faça uma análise dos resultados obtidos na sua implementação.

Questão 4:
Compare os resultados dos três algoritmos implementados, usando os exemplos com 10, 100 e 1000 elementos gerados de forma aleatória e o exemplo já ordenado de 10 elementos.

Questão 5:
Ache uma cota inferior e superior para o problema de linha poligonal através do mecanismo de redução. Assuma que linha poligonal é uma linha que une vértices que não podem se interceptar. Mostre que a redução é linear e que existe um algoritmo ótimo para linha poligonal. Implemente um algoritmo para linha poligonal usando a redução por ordenação e teste para os seus algoritmos de ordenação implementados, usando exemplos com 10, 100 e 1000 elementos gerados de forma aleatória. Analise os resultados da sua implementação.