Regras Gerais para os Labs de CES12

Entrega

Atraso na entrega gera penalidade de 1 ponto/dia

Encontrando pela primeira vez problemas no código fornecido ou enunciado, o aluno pode ganhar um bônus (*além de ajudar a turma, evitando bugs, confusões, atrasos, etc.*), maior ou menor dependendo da criticalidade do problema. Não deixe para a última hora!

O código fornecido pelo professor pode ser copiado ou modificado. No entanto, como deve submeter apenas o diretório src, lembre que os meus testes, configuração de compilação, e bibliotecas continuam os mesmos na minha máquina. Certifique-se que o seu diretório `src` compila com o meu projeto CMake na árvore de diretórios fornecida. Para gerar a sua versão do meu código sem modificá-lo, basta herdar uma das minhas classes, ou até fazer copy-paste, mudar o nome e ter uma versão sua das minhas funções ou classes.

Plágio

Casos de plágio/fraude/cola serão tratados pelo processo formal envolvendo a DOO, a Dival, etc.

É permitido ajudar os colegas em relação à todas as ferramentas utilizadas: linguagens de programação, bibliotecas de terceiros (e.g. STL) e SO (e.g. Linux), ferramentas de compilação (e.g. cmake), teste (e.g. gtest), ou IDEs (e.g. cLion). Inclusive em relação ao entendimento e uso do código fornecido pelo professor.

É permitido (e encorajado) o uso da STL desde que não aproveitem soluções prontas para o propósito do exercício. E.g., não usar stl::sort no lab de ordenação, ou stl::hash_map no lab de hash. *Em caso de dúvida, pergunte*.

Para evitar plágio, não mostre, discuta, ou divulgue o seu código. E.g. procure, discuta e divulgue exemplos na internet da utilização da classe std::vector, ao invés de mostrar como a sua implementação de hash usa std::vector!

Vale ajudar a debugar apenas depois de terminar a sua implementação.

O professor pode aplicar plagiarism checkers nos códigos submetidos.

Testes, Correção e fraudes

Haverá correção visual do código *no mínimo* por amostragem, em relação à corretude da implementação dos Algoritmos.

Mesmo que os testes passem, implementações que não efetivamente implementam os algoritmos requisitados invalidam todas as avaliações dependentes destas implementações: os próprios testes, perguntas, relatórios, etc.

Implementações, mesmo passando nos testes, podem configurar casos de fraude equivalente a cola. E.g., nos casos onde o número de entradas testadas é limitado, responder as respostas certas “hardcoded”, obtidas por outros meios, sem implementar nenhum algoritmo.

Em caso de bug ou incompletude nos testes, o importante é resolver o problema, não é passar nos testes - ou seja, *o teste é um meio e não um fim*. No entanto, soluções erradas que passem nos testes podem ser consideradas após uma avaliação caso a caso.

Implementações que se destaquem da turma por extrema ineficiência serão penalizadas (ou invalidadas se impossibilitar a correção) em relação a todos os testes, comparações, etc. relacionadas à implementação ineficiente. Isto não significa “ser o pior da turma” ou “20% mais lento que a média”. Não é uma competição. Um caso extremo seria implementar um algoritmo cúbico ao invés de um linear que demora minutos enquanto a turma demora 1 segundo. Não esperarei até o seu código terminar.