ACH2023 ALGORITMOS E ESTRUTURAS DE DADOS I

Semestre 2022-2 - Exercício prático 2 – Árvore completa – t.02

Estagiário PAE: Wesley Ramos dos Santos (wesley.ramos.santos@usp.br)

Descrição do EP: A partir do modelo disponibilizado no e-disciplinas (*ep2-modelo.txt*), implementar as seguintes operações para copiar uma ABB criando uma árvore completa.

1. O objetivo do trabalho é implementar de forma correta e completa a função **CopiaCompleta** que cria uma cópia de uma ABB fornecida como entrada de modo que ela passe a ter a **altura mínima possível** para aquela quantidade de elementos *sem deixar de ser ABB*.
2. A função recebe como entrada uma ABB apontada por *raiz contendo chaves inteiras, e deve retornar o ponteiro para uma nova ABB que seja cópia dela com altura mínima, *sem modificar a árvore original*.

   NO* copiaCompleta(NO* raiz)

3. A posição exata de cada elemento na árvore de resposta não é relevante, mas ela deve ser uma árvore ABB. Ou seja, qualquer ABB de altura mínima que contenha exatamete as mesmas chaves de entrada é uma resposta válida.
4. Note que a árvore fornecida não (necessariamente) é uma AVL, e que portanto o uso de rotações AVL não é garantia de que a árvore possa ser transformada em uma árvore completa (que de resto não é o objetivo da rotação AVL mesmo). Sugere-se assim não usar rotações AVL.
5. RESTRIÇÕES DE IMPLEMENTAÇÃO:
   a. Não use nenhum tipo de implementação estática (vetores, listas sequenciais etc.) no seu trabalho. Se necessitar de estruturas auxiliares, use sempre listas ligadas de implementação dinâmica. Uma declaração *typedef NOL* foi incluída no código exemplo caso queira criar algum tipo de lista ligada.
   b. Não use variáveis globais. A função implementada deve definir localmente todas as variáveis e estruturas auxiliares, ou chamar funções auxiliares que o façam também em um escopo local.
6. Não exiba nenhuma mensagem na tela, nem solicite que o usuário pressione nenhuma tecla etc. Apenas implemente a função solicitada.
7. A função *main*() serve apenas para seus testes particulares, e não precisa ser entregue. Caso você prefira mantê-la no corpo do programa, pede-se apenas que *main*() seja a última função do programa, ou seja, que não haja nenhum código abaixo dela.
8. Seu programa será corrigido de forma *automática*, e por isso você não pode alterar as assinaturas da função solicitada, nem os tipos de dados ou especificações (*typedef*) do modelo fornecido.
9. O EP pode ser desenvolvido individualmente. Não tente emprestar sua implementação para outros colegas, nem copiar deles, pois isso invalida o trabalho de todos os envolvidos.
10. 0 programa deve ser compilável no Codeblocks 13.12 sob Windows 10 ou superior. Será aplicado um desconto de até 30% na nota do EP caso ele não seja imediatamente compilável nesta configuração.
11. Programadores JAVA, cuidado: não existe inicialização automática de variáveis em C.

O que/como entregar:

- Preencha a função nroUSP do código disponível.
- Na primeira linha do código, escreva um comentário "//" com o seu nome.
- A entrega será via upload no sistema **antes** do prazo limite da atividade.
- Entregue apenas o código da função principal e das funções auxiliares que ela invoca.
- O nome do arquivo deve ser o seu nro. USP com a extensão .cpp - **favor não compactar**.

Prazos etc.:

O EP deve ser depositado no prazo definido na atividade cadastrada no sistema. Não serão aceitos EPs entregues depois do prazo, independentemente do motivo. Entregas no último dia são assim por conta e risco do aluno, e nenhum tipo de imprevisto de última hora (e.g., problemas de saúde, indisponibilidade de rede etc.) pode ser usado como justificativa para o atraso. O EP é uma atividade para ser desenvolvida ao longo de várias semanas, não no último dia da entrega.

É responsabilidade do aluno que fez o *upload* do arquivo verificar se o mesmo foi corretamente recebido pelo sistema. Atrasos/falhas na submissão invalidam o trabalho realizado. Certifique-se também de que a versão entregue é a correta. Não serão aceitas substituições.

Critérios de avaliação:

A função será testada com uma série de chamadas repetidas e consecutivas, fornecendo diversas árvores não vazias contendo inteiros positivos como entrada. É assim importante assegurar que o seu programa funciona desta forma (por exemplo, chamando-o dentro de um laço *for*), e não apenas para um teste individual. Um teste é considerado correto se o resultado da soma for exatamente como o esperado, ou incorreto em caso contrário. Erros de alocação de memória ou compilação invalidam o teste, assim como a ausência de funções auxiliares necessárias para a execução do programa.

Este EP deve ser desenvolvido obrigatoriamente por *todos* os alunos de AED1. Sua nota é parte integrante da 3ª. avaliação e *não é* passível de substituição.