ACH2023 ALGORITMOS E ESTRUTURAS DE DADOS I

Semestre 2020-2 - Exercício prático 3 – Organização de árvore de busca binária – t.02

Estagiário PAE: Thais Souza (thais.donega@usp.br)

Descrição do EP: A partir do modelo disponibilizado no Tidia (*ep3-modelo.txt*), implementar as seguintes operações para organização de uma árvore de modo que ela se torne uma ABB:

1. O objetivo do trabalho é implementar de forma correta e completa a função **organizar** em uma árvore que pode ser uma ABB, ou é "quase" uma ABB, tendo no máximo um único elemento fora da ordem esperada.
2. A função recebe como entrada uma árvore apontada por **raiz (a passagem do ponteiro é por referência porque o ponteiro raiz pode ser modificado) contendo chaves inteiras, e deve detectar se há ou não um elemento fora da ordem. Se houver, o elemento deve ser excluído. Se não houver, nada deve ser modificado.

   void organizar(NO raiz)**
3. Note que a árvore resultante, com ou sem organização, é sempre uma ABB perfeitamente ordenada.
4. Caso especial: toda árvore terá no máximo uma chave incorreta. Entretanto, no caso extremo, pode haver situações em que não é possível determinar qual é este elemento. Isso pode ocorrer, por exemplo, em uma árvore desordenada contendo apenas duas chaves. Nestes casos, o elemento a ser corrigido deve ser sempre *o que estiver incorreto e for o mais próximo possível da raiz.*
5. RESTRIÇÕES DE IMPLEMENTAÇÃO:
   a. Não use nenhum tipo de ED estática (vetores, listas sequenciais etc.) no seu trabalho. Se necessitar de estruturas auxiliares, use sempre listas ligadas de implementação dinâmica.
   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. Não entregue soluções incompletas ou não compiláveis, pois elas só atrasam a avaliação dos trabalhos da turma.
10. Não tente emprestar sua implementação para outros colegas, nem copiar deles, pois isso invalida o trabalho de todos os envolvidos.
11. 0 programa deve ser compilável no Codeblocks 13.12 sob Windows 7 ou superior. Será aplicado um desconto de até 30% na nota do EP caso ele não seja imediatamente compilável nesta configuração.
12. Programadores JAVA, cuidado: não existe inicialização automática de variáveis em C.

O que/como entregar:

- A entrega será via upload no sistema Tidia.
- Entregue apenas o código da função principal e das funções auxiliares que ela invoca.
- A extensão do arquivo pode ser .c ou .cpp - **favor não compactar**.
- Preencha a função nroUSP) do código disponível no Tidia para que você seja identificado.

Prazos etc.:
O EP deve ser depositado no prazo definido na atividade cadastrada no sistema Tidia. 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.

O sistema Tidia envia um email de confirmação da submissão. 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 *antes* do prazo final. 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.