ACH2023 ALGORITMOS E ESTRUTURAS DE DADOS I

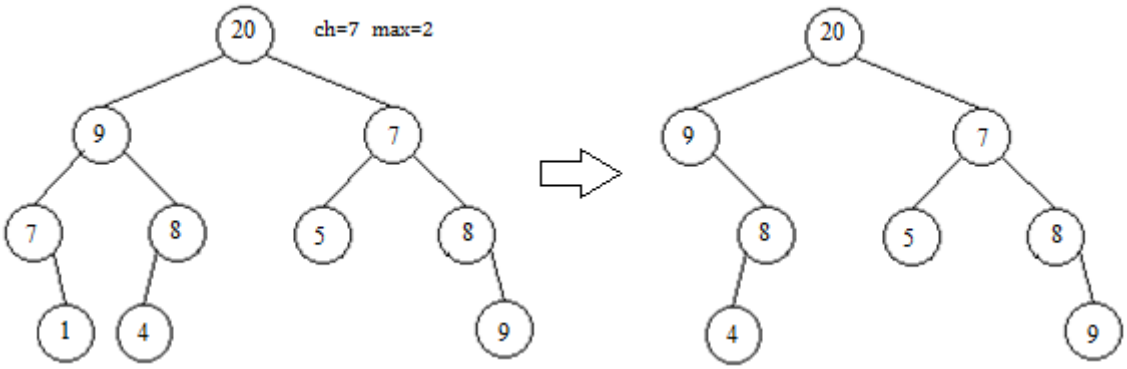
Semestre 2013-1 - Exercício prático 2 – Exclusão de subárvores em Árvore binária

Descrição do EP: A partir do projeto exemplo disponibilizado no COL (*projetoArvore.dev*), editar o módulo *trabalho.cpp* implementando as seguintes operações para manipulação de uma binária comum (i.e., não é uma ABB) contendo chaves inteiras positivas, possivelmente repetidas.

- O projeto exemplo fornecido contém uma função *main.cpp* que pode ser usada para execução e teste do seu programa, *arvore.h* contendo os tipos de dados manipulados e outras especificações globais, e o módulo *trabalho.cpp* onde deve ser realizado o trabalho propriamente dito.
- O trabalho consiste em implementar de forma correta e completa a função **void excluirSubarvores(NO **raiz, int ch, int max)**. Em especial, será avaliada a complexidade de espaço da sua implementação, conforme descrito a seguir.

- A função recebe como entrada uma árvore não ordenada *raiz*, uma chave *ch* e um inteiro *max*.
- O objetivo da função é excluir todos os elementos das subárvores cuja raiz possui valor *ch*, e que possuam até *max* descendentes.
- A saída da função é a árvore passada por referência, devidamente atualizada.

- Para evitar que uma exclusão afete o resultado de outras, as exclusões devem ser consideradas em ordem de nível, começando pela raiz, depois considerando os filhos imediatos da raiz etc.
- Exemplo: dada a árvore abaixo e supondo-se *ch*=7 e *max*=2, seria excluída apenas a subárvore de raiz 7 do lado esquerdo, mas não a do lado direito (que possui mais de *max=2* descendentes).



- A árvore passada como parâmetro pode eventualmente ser vazia, assim como o valor de *ch* pode não ocorrer na estrutura. Além disso, se a raiz da árvore for excluída ela deve se tornar NULL.
- Não use *malloc* ou *free* na sua implementação. A utilização destes comandos invalida o seu EP. Para criar e destruir nós, você deve chamar as funções *criar* e *destruir* já definidas em trabalho.cpp, as quais invocam as respectivas funções *malloc* e *free*. O objetivo desta restrição é avaliar a quantidade de nós criados e destruídos pelo seu programa, ou seja, sua complexidade de espaço.
- Não use variáveis globais. A função implementada deve definir localmente todas as variáveis e estruturas auxiliares, ou chamar funções auxiliares que o façam também dentro de um escopo local. O objetivo desta restrição é garantir que sua função possa ser testada de forma independente do seu código de teste.
- Não exiba nenhuma mensagem na tela, nem solicite que o usuário pressione nenhuma tecla etc. Apenas implemente a função solicitada.

- A função implementada deve estar inteiramente contida no módulo *trabalho.cpp*. Note no entanto que para testar a sua implementação e garantir sua correção você provavelmente terá de criar várias outras funções auxiliares (e.g., entrada de dados, inserção, exibição etc.) que não fazem parte do trabalho e que não serão avaliadas.
- Observe que tudo que for necessário para executar a função solicitada deve obrigatoriamente estar em *trabalho.cpp*, ou seu EP estará incompleto.
- Depois de tudo pronto, certifique-se de que seu programa funciona corretamente **quando chamado a partir de main()**. É um erro comum desenvolver e testar a função diretamente em trabalho.cpp esquecendo que a função tem que retornar uma resposta correta para main().
- Certifique-se também de que seu programa retorna o valor correto quando chamado várias vezes de forma consecutiva dentro de um laço for(), pois **é assim que será testado**; funções bem projetadas devem responder corretamente a todas situações.
- Este trabalho pode ser desenvolvido *individualmente* ou pelas *mesmas duplas* do EP anterior. Duplas já formadas podem ser desfeitas, mas novas duplas não são aceitas.
- Não tente emprestar sua implementação para outros colegas/duplas, nem copiar deles, pois isso invalida o trabalho de **todos** os envolvidos. Emprestar o trabalho "apenas para o colega dar uma olhada" é uma forma comum de reprovação na disciplina.
- Além da função acima, solicita-se que o aluno complete as funções *aluno1*, *aluno2*, *nrousp1* e *nrousp2,* também presentes no módulo *trabalho.cpp* para fins de identificação dos autores. Para trabalhos individuais, deixe as informações do segundo autor em branco, mas não remova as funções *aluno2* e *nrousp2* do código.
- 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 em *arvore.h.*
- 0 programa deve ser compilável no Dev-C++ versão 4.9.9.2. sob Windows 7 ou superior.
- Caso o trabalho seja desenvolvido em uma versão anterior do Windows, ou em Linux, é **fundamental** que seja testado exaustivamente na plataforma solicitada acima. O sistema de gerenciamento de memória das versões recentes do Windows é menos flexível que nas versões anteriores, e erros de programação que seriam desconsiderados, por exemplo, pelo Windows-XP, costumam provocar erro de alocação de memória nas versões mais recentes do sistema. Não invalide seu trabalho por causa deste detalhe.

O que entregar:
Apenas o arquivo *trabalho.cpp* extraído do projeto implementado, observando que ele deve conter todas as rotinas necessárias para a execução da função solicitada. A entrega será realizada via sistema COL **até o horário de início da SUB**. EPs entregues após este horário serão desconsiderados. Não compacte seu arquivo, tome cuidado para enviar a versão correta (e não um EP anterior ou um arquivo vazio), e atenção ao horário.

Critérios de avaliação:
O programa será testado com uma série de 5 chamadas consecutivas (dentro de um laço for), atribuindo-se até 2 pontos para cada resposta correta. Uma resposta é considerada correta se, além de resultar em uma árvore consistente e que pode ser percorrida normalmente (critério eliminatório), tiver (a) efetuado a exclusão de todas as subárvores esperadas (1.6 pontos) e (b) tiver feito o número correto de chamadas à função *destruir* (0.4 pontos). Eventuais descontos por mal funcionamento etc. serão tratados caso a caso.

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