ACH2023 ALGORITMOS E ESTRUTURAS DE DADOS I

Semestre 2013-1 - Exercício prático 1

Exclusão do K-ésimo elemento "reverso" em Listas Ligadas

Descrição do EP: A partir do projeto exemplo disponibilizado no COL (*projetoListas.dev*), editar o módulo *trabalho.cpp* implementando as seguintes operações para manipulação de listas ligadas de inteiros:

1. O projeto exemplo fornecido contém uma função *main.cpp* que pode ser usada para execução e teste do seu programa, o módulo *listas.h* contendo os tipos de dados manipulados e outras especificações globais, e o módulo *trabalho.cpp* onde deve ser realizado o trabalho propriamente dito.
2. O objetivo do trabalho é implementar de forma correta e completa a função **removerK** utilizando apenas listas ligadas simples, não circulares, sem cabeça e sem nó sentinela como estrutura de armazenamento. Em especial, será avaliada a complexidade de espaço da sua implementação, conforme descrito a seguir.
3. A função recebe como entrada uma lista de inteiros e um valor k. O objetivo da função é remover o k-ésimo elemento da lista (se houver), *contado do fim para o começo da estrutura*. O último elemento da lista é excluído se k=1, o penúltimo quando k=2 etc.
4. Note que o valor de k pode ser inválido, e/ou que a lista fornecida pode ser vazia.
5. A assinatura da função é a seguinte:

   void removerK(NO *p, int k)

   sendo p o ponteiro de início da lista fornecida como entrada.
6. Restrições de implementação:

   (a) *Não use nenhum vetor na sua implementação*. Estruturas auxiliares de implementação dinâmica podem ser utilizadas livremente, mas caso haja uso de qualquer tipo de estrutura estática o EP será considerado inválido. O objetivo desta restrição é estimulá-lo a praticar o uso de estruturas de implementação dinâmica.

   (b) Não use *malloc* ou *free* na sua implementação. A utilização destes comandos invalida o seu EP. Para criar e destruir nós, você deve obrigatoriamente 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, verificar a complexidade de espaço do seu algoritmo.

   (c) 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.
7. Não exiba nenhuma mensagem na tela, nem solicite que o usuário pressione nenhuma tecla etc. Apenas implemente a função solicitada.
8. 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, exibição etc.) que não fazem parte do trabalho e que não serão avaliadas.
9. Observe que tudo que for necessário para executar a função solicitada deve obrigatoriamente estar em *trabalho.cpp*, ou seu EP estará incompleto.

10. Além da função acima, solicita-se que o aluno complete as funções *aluno1*, *aluno2*, *nrousp1* e *nrousp2* e *turma,* 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.
11. 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 *listas.h.*
12. 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 as listas resultantes devem ser retornadas e percorridas em main(), o que nem sempre funciona da forma esperada dependendo da técnica de alocação de memória empregada. Portanto, percorra sua lista de resposta em main() para saber se o programa realmente funciona.
13. O EP pode ser desenvolvido *individualmente ou em duplas* desde que previamente confirmadas em sala de aula *até o final do mês de abril*. Após esta data, somente trabalhos individuais serão considerados.
14. Cada dupla confirmada receberá um número que deve ser utilizado para fazer o *upload* do trabalho no sistema COL.
15. Não tente emprestar sua implementação para outros colegas, nem copiar deles, pois isso invalida o trabalho de todos os envolvidos. Repito: não empreste seu trabalho aos seus colegas.
16. 0 programa deve ser compilável no Dev-C++ versão 4.9.9.2. sob Windows Vista ou 7.
17. 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 do que o das 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.
18. Programadores JAVA, cuidado: não existe inicialização automática de variáveis em C.

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 a data e hora de inicio da P1. EPs entregues após este horário serão desconsiderados, além de invalidar a P1.

Critérios de avaliação:
A função será testada com uma série de 10 chamadas consecutivas, valendo 1,0 ponto cada, sendo:

0,5 ponto pelo retorno do valor correto.
Se o valor estiver correto, + 0,2 ponto se a quantidade de nós criados for compatível com o problema.
Se o valor estiver correto, + 0,3 ponto se foram destruídos todos os nós temporários e o k-ésimo nó,
e nada além destes.

Este EP deve ser desenvolvido obrigatoriamente por *todos* os alunos de AED1. Sua nota faz parte da 1ª. avaliação da disciplina. A nota do EP **não é passível de substituição**, e é a principal causa de reprovação na disciplina. Caso o EP não seja entregue, isso implica nota zero na 1ª. avaliação independentemente do resultado na P1 ou SUB.