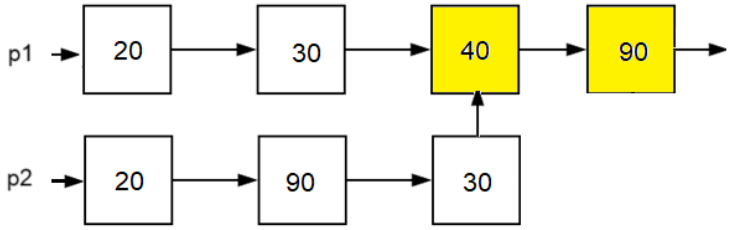
ACH2023 ALGORITMOS E ESTRUTURAS DE DADOS I

Semestre 2023-2 - Exercício prático 1 – Remoção de elementos compartilhados

Responsável - Ivandré Paraboni

1. O objetivo do trabalho é implementar de forma correta e completa as seguintes funções utilizando apenas listas ligadas simples como entrada, usando o modelo de código disponibilizado no sistema e-disciplinas.
2. A função a ser implementada recebe como entrada duas listas ligadas p1 e p2 não vazias, e que compartilham zero ou mais nós em seu segmento final. O exemplo a seguir ilustra uma situação em que duas listas possuem um segmento final composto de dois elemento compartilhados (em amarelo).



3. As listas possuem chaves aleatórias do tipo inteiro, sem ordem e com possível repetição. Assim, o compartilhamento a ser considerado é referente ao uso do mesmo segmento final de nós acessíveis tanto via p1 quanto p2, em amarelo, e não tem relação com o valor da chave em si. Por exemplo, na ilustração acima existe apenas um nó de valor 90 comum a ambas as listas, embora a lista p2 possua também um segundo elemento de chave 90 que não é acessível a partir de p1, e que portanto não é parte do segmento compartilhado em amarelo.
4. O objetivo do trabalho é **remover o segmento compartilhado** (se houver) ao final das listas, tornando-as completamente disjuntas e **tornando-as circulares**. Assim, se uma lista for completamente coincidente com a outra, ambas ficarão vazias, pois todos elementos compartilhados devem ser removidos.
5. No caso acima, as listas resultantes seriam p1={20,30} e p2={20,90,30}, ambas em formato circular.
6. Lembre-se de que a resposta consiste de duas listas circulares totalmente separadas. Se sua resposta terminar em NULL, ou se restar algum nó compartilhado, a resposta estará errada.
7. As duas listas são passadas na forma de ponteiros por referência para que seus inícios possam ser modificados se necessário. A assinatura da função é a seguinte:

```
void removerCompartilhados(NO* *p1, NO* *p2);
```

8. Os ponteiros fornecidos como entrada (*p1 e *p2) são o início de listas, que nunca estão vazias. Nesta passagem de parâmetros por referência, a primeira chave da lista p1 pode ser acessada usando (*p1)->chave, por exemplo.
9. Restrições de implementação:

   (a) *Não use nenhum vetor na sua implementação*. Estruturas auxiliares de implementação dinâmica podem ser utilizadas livremente, mas o uso de implementação estática invalida o trabalho.

   (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.

   (c) Não exiba nenhuma mensagem na tela, nem solicite que o usuário pressione nenhuma tecla etc.

   (d) Não utilize código disponibilizado *online* ou em outras fontes, nem ferramentas de geração de código automático.

10. A função main() serve apenas para seus testes particulares, e não deve ser incluída na sua entrega.
11. O EP pode ser desenvolvido **individualmente ou em duplas pré-cadastradas no link abaixo** dentro do prazo a ser estipulado em aula, sem compartilhamento de código de espécie alguma. Não tente emprestar sua implementação para outros colegas, nem copiar deles, pois isso invalida o trabalho de todos os envolvidos. Seu trabalho é confidencial, e não pode ser compartilhado ou disponibilizado de forma alguma.
12. O link para cadastro de grupos (individuais ou em dupla) é o seguinte:

    https://docs.google.com/spreadsheets/d/1gR0S9nqxRn4g0IAFIs0FVur5-6RralHKynvMd6lv304/edit?usp=sharing
13. 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.
14. 0 programa deve ser compilável no ambiente Windows com Codeblocks 13.12 ou superior. Se você não conseguiu fazer o programa compilar sem erros dentro do prazo, não entregue. Não é previsto nenhum tipo de avaliação além dos testes de execução, o que pressupõe um programa compilável e funcional.

O que/como entregar:

- A entrega será via upload no sistema **antes** da data estipulada.
- Entregue apenas um arquivo texto com o código da função principal e funções auxiliares que ela invoca.
- O nome do arquivo deve ser seunome.cpp - **favor não compactar** – onde "seunome" deve ser substituído pelo primeiro nome do responsável por entregar o código.
- Preencha as funções nroUSP do código exemplo disponível para que você seja identificado.

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 responsável, 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 nos últimos dias antes 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. Após o *upload*, verifique se você consegue abrir o arquivo depositado, e certifique-se de que é a versão correta do programa e que não está corrompido.

Critérios de avaliação:

A função será testada com uma série de chamadas repetidas e consecutivas, com diversas listas de 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 for exatamente como o esperado, ou incorreto em todo e qualquer caso divergente. 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 1ª. avaliação e *não é* passível de substituição.