Universidade Federal de São Carlos – Departamento de Computação
Algoritmos e Estruturas de Dados I
Prof. Alexandre L. M. Levada

Atividade Avaliativa 4
Listas encadeadas e duplamente encadeadas

A entrega de apenas um documento com os textos dos códigos, sem um link para execução online, acarretará em desconto na nota.

Para a resolução dos exercícios, é preciso implementar um TAD ou classe que represente a estrutura de dados em questão (node, lista encadeada, lista duplamente encadeada)

📘 1. Exercício: A Missão do Guardião das Memórias

No mundo digital de Memória Eterna, as lembranças de cada habitante são armazenadas em listas encadeadas, onde cada nó representa uma memória e aponta para a próxima. Um misterioso vírus chamado "Rewind" inverte a ordem dessas memórias, fazendo com que as pessoas se lembrem dos eventos do fim para o começo.

Você foi convocado como Guardião das Estruturas para estudar e implementar a operação de inversão da memória, restaurando ou alterando conscientemente a ordem das lembranças.

Estrutura da Memória (Lista Encadeada)

Cada nó da lista é representado por:

```
Node {
   id: inteiro
   memoria: string
   prox: Node
}
```

a) Implemente uma função inverterLista(L) que inverta uma lista encadeada in place, ou seja, sem alocar nova memória para os nós (não deve ser utilizada estrutura auxiliar). Sua função deve retornar a nova cabeça da lista invertida. Calcule a complexidade da função.

b) Implemente uma função inverterLista_S(L) que inverta uma lista encadeada utilizando uma pilha como estrutura de dados auxiliar. Calcule a complexidade da função.

c) Implemente uma função recursiva inverterLista_R(L) que inverta uma lista encadeada usando recursão. Calcule a complexidade da função.

Teste cada uma das funções implementadas em uma lista encadeada com o seguinte conjunto de memórias afetado pelo vírus “Rewind”:

(12, vida adulta) -> (11, trabalho) -> (10, faculdade) -> (9, ensino médio) -> (8, amizade) -> (7, adolescência) -> (6, ensino fundamental) -> (5, natal) -> (4, praia) -> (3, infância) -> (2, pai) -> (1, mãe)

📘 2. Exercício: Os Arquivistas de Chronos

No reino de Chronos, há dois Arquivistas responsáveis por manter registros de eventos em listas encadeadas, cada uma ordenada cronologicamente (em ordem crescente). Com o aumento dos acontecimentos, o Conselho ordenou que os Arquivistas unissem suas listas em um único arquivo histórico — sem perder a ordenação e sem duplicar registros de tempo idêntico.

Como Guardião das Estruturas, sua missão é implementar a função que faz essa intercalação ordenada de duas listas encadeadas, representando os arquivos dos Arquivistas.

Definição da Estrutura:

```
struct Node {
   valor: inteiro
   prox: Node
}
```

Cada lista individual contém valores ordenados em ordem crescente.

a) Implemente uma função intercalarListas(L1, L2) que:
Recebe como entrada duas referências L1 e L2 para o início das listas encadeadas.
Retorna o ponteiro para o início de uma nova lista ordenada, que contenha os nós intercalados de L1 e L2.
Não deve alocar novos nós – apenas rearranjar os ponteiros existentes.
Elimina valores duplicados, mantendo apenas uma ocorrência.

Exemplo:
L1: $1 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow \text{nil}$
L2: $2 \rightarrow 3 \rightarrow 6 \rightarrow 8 \rightarrow \text{nil}$
Resultado esperado:
$1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow \text{nil}$

Teste sua função nas listas a seguir:
L1: $1 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow \text{nil}$
L2: $2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 10 \rightarrow 12 \rightarrow \text{nil}$

b) Calcule matematicamente a complexidade da função desenvolvida no item anterior.

📘 3. Exercício: O Arqueólogo dos Dados

No Arquivo Antigo de Computália, as informações históricas foram gravadas como sequências de eventos repetidos, representadas em uma lista duplamente encadeada. Para organizar e analisar melhor essas relíquias, os arqueólogos precisam produzir uma versão compactada, onde cada evento aparece apenas uma vez, junto com sua frequência de ocorrência.

Você, como Arqueólogo dos Dados, recebeu a missão de desenvolver um algoritmo que leia uma lista duplamente encadeada L, cujos nós armazenam números inteiros possivelmente repetidos, e produza uma nova lista C, também duplamente encadeada, onde cada número aparece apenas uma vez, junto com a quantidade de vezes que ocorre em L, na ordem crescente das chaves.

Estrutura dos Nós
Você deverá trabalhar com as seguintes estruturas:

```
struct NodeL {
   valor: inteiro
   prev: NodeL
   prox: NodeL
}

struct NodeC {
   key: inteiro
   count: inteiro
   prev: NodeC
   prox: NodeC
}
```

a) Implemente uma função compactarLista(L) que:

Percorre a lista L (de inteiros), contabilizando as ocorrências de cada número.
Cria e retorna uma nova lista C, onde cada nó possui a chave e sua contagem.
A lista C deve estar ordenada por valor de chave em ordem crescente.
Utilize alocação dinâmica dos nós (sem reaproveitar os nós de L).

Por exemplo, seja a lista duplamente encadeada L a seguir:

4 ↔ 2 ↔ 1 ↔ 4 ↔ 2 ↔ 3 ↔ 1 ↔ 3 ↔ 5 ↔ 5 ↔ 2 ↔ 2 ↔ 3 ↔ 2 ↔ 4 ↔ 1 ↔ 3 ↔ 5 ↔ NIL

Então, sua lista compactada C deve ser:

(1, 3) ↔ (2, 5) ↔ (3, 4) ↔ (4, 3) ↔ (5, 3) ↔ NIL

b) Calcule matematicamente a complexidade da função desenvolvida no item anterior.

📘 4. Exercício: A Lista Circular do Castelo

No castelo de Algorithmland, há uma lista circular mágica de cavaleiros esperando para participar de um torneio. Essa fila é mantida por uma lista duplamente encadeada circular, onde cada cavaleiro pode ir para frente ou para trás na fila e, ao chegar novamente ao final, retorna ao início.

Periodicamente, o Rei decreta rodadas de reordenação: cavaleiros com números ímpares pulam para a frente da fila, enquanto os pares permanecem onde estão. Esse processo é repetido várias vezes, e a ordem dos cavaleiros muda de forma dinâmica.

Você, como guardião da lógica do castelo, deve implementar uma função que simula uma rodada de reordenação segundo essas regras.

Estrutura da Lista Circular Duplamente Encadeada

```
struct Node {
   id: inteiro            // identificador do cavaleiro
   prev: Node
   prox: Node
}
```

A lista é circular, ou seja:

O primeiro elemento aponta para o último como ant.
O último aponta para o primeiro como prox.

Implemente uma função reorganizarLista(L) que:

Percorre a lista circular duplamente encadeada.
Remove todos os nós com valores ímpares da posição original.
Insere esses nós na frente da lista, mantendo sua ordem relativa original entre eles.
Os nós pares permanecem em suas posições originais.
A lista deve continuar sendo circular e duplamente encadeada.

Exemplo

Entrada original:

[4] ⇄ [3] ⇄ [2] ⇄ [7] ⇄ [8] ⇄ [5] ⇄ [6] ⇄ [9] ⇄ (volta para o [4])

Após reorganização (ímpares vão para frente):

[3] ⇄ [7] ⇄ [5] ⇄ [9] ⇄ [4] ⇄ [2] ⇄ [8] ⇄ [6] ⇄ (volta para o [3])

Execute a função com a entrada acima e verifique o resultado obtido.

OBS: As implementações dessa atividade devem ser entregues na forma de link para o OnlineGDB ou Google Colab para que eu possa executar os programas.

"You are a piece of the puzzle of someone else's life. You may never know where you fit, but others will fill the holes in their lives with pieces of you."
-- Bonnie Arbon