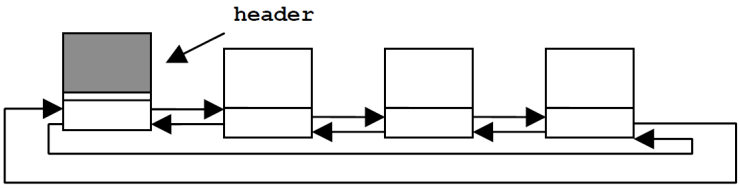
UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação - ICMC
SCC0202 – Algoritmos e Estrutura de Dados I - 2º Sem /2022
Prof. Rudinei Goularte, Marcelo Manzato
PAEs: Wan Song, Fernanda Marana

Lista de Exercícios Número 4

1. Uma lista circular duplamente encadeada com nó de cabeçalho (header) pode ser esquematizada como na figura abaixo. Note que o último nó aponta para o primeiro (e não para NULL). Suponha que o header guarde o número de elementos da lista. Implemente uma lista assim, com as seguintes operações:

- cria (L)
- IsEmpty(L)
- IsFull(L)
- esta-na-lista (L , X) //verifica se X está na lista L
- insere-a-direita-de (L, X, Y) //insere X a direita de Y na lista L



Adicional: Faça considerações sobre o impacto do uso de uma lista ordenada nas funções de inserção, remoção e busca por elementos? Elas ficam mais caras ou mais baratas?

2. Quais são as utilidades de um nó de cabeçalho. Exemplifique.

3. O que é uma lista linear? E uma lista não linear? Dê exemplos e justifique essa nomenclatura (linear vs. não linear).

4. Especifique um problema que é melhor de ser resolvido com uma representação estática e sequencial e outro que seja melhor resolvido com uma representação dinâmica e encadeada. Justifique.

5. Se você tem que escolher entre uma representação por lista encadeada ou uma representação usando posições contíguas de memória para um vetor, quais informações são necessárias para você selecionar uma representação apropriada? Como estes fatores influenciam a escolha da representação?

6. Construa um método que recebe uma lista encadeada de números inteiros e retorna uma lista sem repetições, ou seja, uma lista onde cada número apareça apenas uma vez. Exemplo:

   12 5 -7 8 5 9 12 1 8 → 12 5 -7 8 9 1

7. Implemente uma rotina recursiva para calcular o tamanho de uma lista estática e encadeada – implica em não usar o atributo *tamanho* do TAD Lista. Implemente também uma versão para lista dinâmica e encadeada.

8. Implemente uma função **não** recursiva para verificar se duas listas encadeadas e dinâmicas L1 e L2 são iguais.

9. Implemente uma função recursiva para verificar se duas listas encadeadas e dinâmicas L1 e L2 são iguais.

10. Dadas duas listas encadeadas e dinâmicas L1 e L2, sem elementos repetidos, implemente a operação INTER, que cria uma terceira lista L3 com a intersecção entre as duas listas, também sem elementos repetidos.

11. Suponha que cadeias de caracteres são armazenadas em listas encadeadas com um caractere por célula. Escreva função análogas a strlen, strcmp, strcpy e strcat.

12. Escreva uma função que recebe o endereço (do primeiro nó) de uma lista encadeada não-circular (ou seja, o último nó aponta para NULL) e retira da lista todos os nós cujo item vale 0. Faça duas versões: na primeira, o primeiro nó da lista é um nó-cabeça (o valor de item é irrelevante); na segunda, a lista não tem nó-cabeça (sua função deve devolver algo nesse caso?).

13. Os nós de uma lista encadeada são usualmente alocados através da função malloc. Em geral, quando um nó é retirado da lista ele pode ser liberado, por meio da função free, para que o correspondente espaço de memória possa ser reaproveitado. Escreva uma função que, ao receber o endereço de uma lista não-circular (o último nó aponta para NULL), libera todos os nós da lista.

14. Escreva uma função que receba uma lista encadeada e devolve o endereço de um nó que esteja o mais próximo possível do meio da lista. Faça isso sem contar explicitamente o número de nós da lista.

15. A função abaixo recebe uma lista encadeada `ll` e um número `y`. Ela deveria devolver o endereço do primeiro `nó` da lista cujo `item` seja igual a `y`; se tal `nó` não existe, a função deveria devolver `NULL`.

```
link procura (int y, link ll) {
   int achou;
   achou = 0;
   while (ll != NULL && achou == 0) {
       if (ll->item == y) achou = 1;
       ll = ll->next;
   }
   if (achou == 1) return ll;
   else return NULL;
}
```

A função funciona como deveria? Escreva uma nova versão da função que seja simples, curta e correta.

Fim da lista 2.