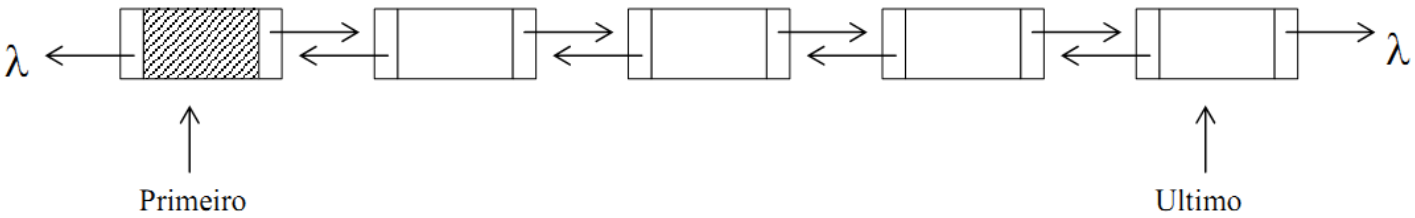
Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Algoritmos e Estruturas de Dados II (Turmas M, N, W, F)
1º Semestre de 2012
Profs. Camilo Oliveira, Gisele Pappa, Ítalo Cunha, Loïc Cerf, William Schwartz

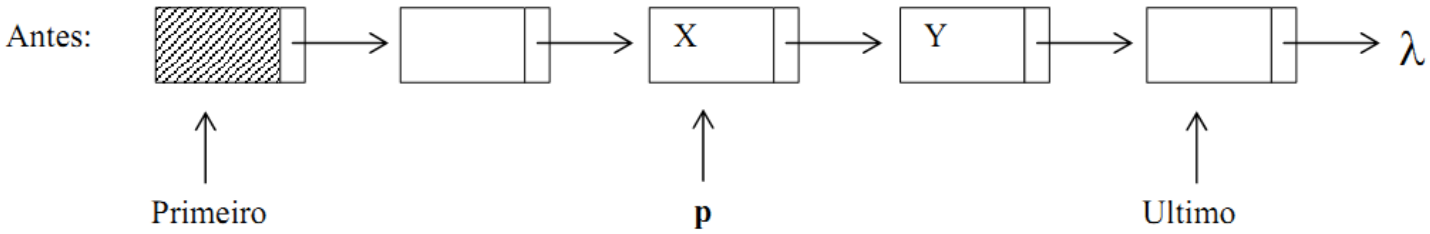
2ª Lista de Exercícios

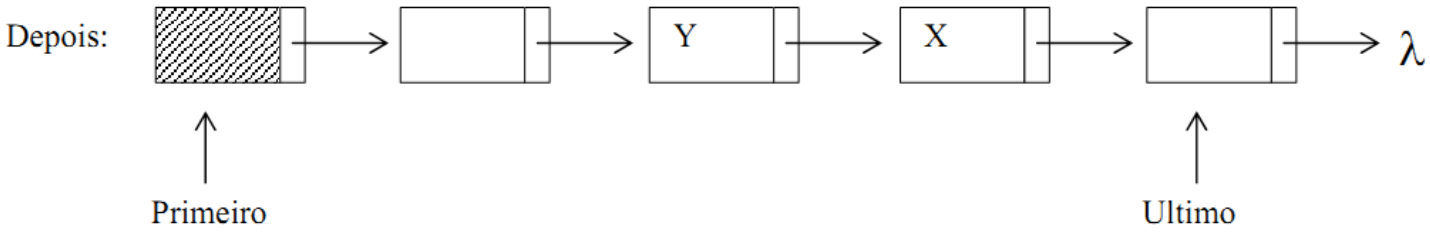
1. Utilizando a implementação de listas com alocação sequencial (arranjos), implemente um procedimento para inserir um item em uma determinada posição da lista. Qual é a ordem de complexidade do seu procedimento?
2. Um problema que surge com frequência na manipulação de listas lineares implementadas através de apontadores é "andar para trás" na lista, ou seja, percorrê-la no sentido inverso dos apontadores. Uma solução para isso é implementar uma lista duplamente encadeada, em que cada célula também possui um apontador para a sua antecessora, como mostrado na figura abaixo:



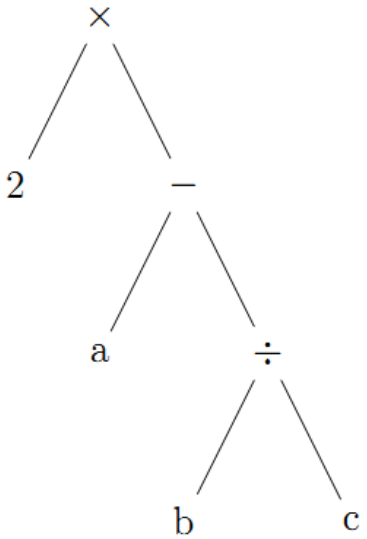
a) Declare os tipos necessários para a implementação dessa lista
b) Escreva um procedimento para retirar uma célula apontada por um apontador p
c) Escreva um procedimento para inserir um elemento na primeira posição da lista (logo após a célula cabeça).

3. Considere uma pilha P vazia e uma fila F não vazia. Utilizando apenas os testes de fila e pilha vazias, as operações *Enfileira*, *Desenfileira*, *Empilha*, *Desempilha*, e uma variável *aux* do tipo de dados contidos na lista, escreva um programa que inverte a ordem dos elementos da fila.
4. Considere a implementação de listas encadeadas utilizando apontadores vista em sala. Escreva um procedimento *Troca(struct lista *Lista, struct no *p)* que, dado um apontador para uma célula qualquer (p), troca de posição essa célula com a sua célula seguinte da lista, como mostrado na figura abaixo. (Obs. Não vale trocar apenas o campo item! Você deverá fazer a manipulação dos apontadores para trocar as duas células de posição). Não esqueça de tratar os casos especiais.



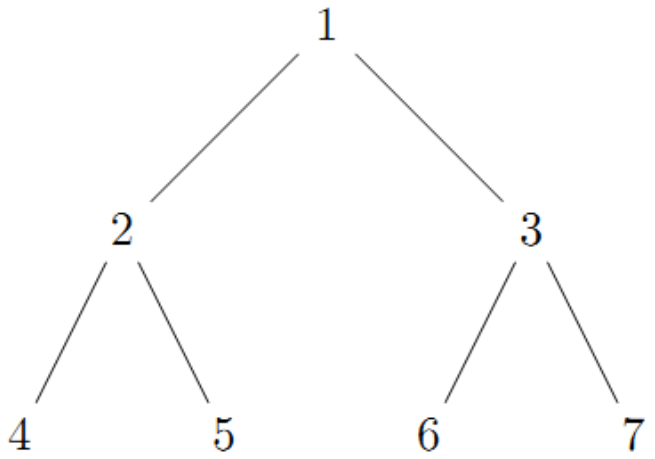


5. Utilizando as operações de manipulação de pilhas vistas em sala, uma pilha auxiliar e uma variável *aux* do tipo de dados contidos na pilha, escreva um procedimento que remove um item com chave **c** de uma posição qualquer de uma pilha. Note que você não tem acesso à estrutura interna da pilha (topo, item, etc), apenas às operações de manipulação.
6. Escreva dois procedimentos (um usando a implementação por arranjos (alocação sequencial) e outro usando a implementação por apontadores) para remover de uma lista encadeada um elemento com uma chave específica (passada como parâmetro): *Remove(struct lista *Lista, int c)*. Qual é a ordem de complexidade dos seus procedimentos?
7. Considere a implementação de filas usando arranjos "circulares" (alocação sequencial). Escreva um procedimento FuraFila(struct fila *fila, int x) que insere um item na primeira posição da fila. O detalhe é que seu procedimento deve ser **O(1)**, ou seja, não pode movimentar os outros itens da fila.
8. Expresse as seguintes equações usando árvores. Cada operador deve ser nó interno que terá os operadores como filhos. Por exemplo, a equação 2(a-b/c) é representada pela seguinte árvore:



a) a + b + 5c
b) 7a/c - c
c) e^{-x^2}

9. Mostre o resultado de caminhamento pré-ordem, central e pós-ordem na árvore abaixo.



10. No máximo quantas chamadas recursivas podem ser feitas à função pré-ordem numa árvore binária? Quantas chamadas recursivas à função pré-ordem podem ser feitas numa árvore binária completa?

11. Implemente um método para inserir um elemento numa árvore binária de pesquisa onde cada nó contém um ponteiro para seu pai, como a seguir:

```
struct arvore {
   struct arvore *pai;
   struct arvore *esquerda;
   struct arvore *direita;
   int dado;
};
```

Seu método deve ter o seguinte protótipo:

```
void insere(struct arvore *raiz, int *dado);
```