UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E COMPUTAÇÃO

SCC 202 - Algoritmos e Estruturas de Dados I

Lista 3 de Exercícios
(Listas Seqüenciais, Encadeadas, Duplamente Encadeadas e Circulares)

1. Crie o Tipo Abstrato de de Dados Lista Linear Ordenada (pelo campo chave), tal que sua **implementação Sequencial**, inclua procedimentos/funções para:

   (a) Verificar se uma lista está ordenada ou não (a ordem pode ser crescente ou decrescente).

   (b) Fazer uma cópia da lista L_1 em uma outra lista L_2.

   (c) Fazer uma cópia da lista L_1 em outra L_2, eliminando os elementos repetidos.

   (d) Inverter uma lista L_1 colocando o resultado em L_2.

   (e) Intercalar duas listas, L1 e L_2, gerando uma lista L_3. Considere que L_1, L_2 e L_3 estão ordenadas.

   (f) Dada uma lista L1, gerar uma lista L_2 onde cada registro contém dois campos de informação: *elem* contém um elemento de L_1 e *count* contém quantas vezes este elemento apareceu em L_1.

   (g) Assumindo que os elementos de uma lista L_1 são inteiros positivos, fornecer os elementos que aparecem o maior e o menor número de vezes (forneça os elementos e o número de vezes correspondente).

2. Uma lista linear circular é mantida num array C [0..*n*-1] com C e F com as mesmas funções de começo e fim em filas circulares.

   (a) Obtenha uma fórmula em termos de C, F e *n* para o número de elementos da lista.

   (b) Escreva um algoritmo para eliminar o k-ésimo elemento da lista.

   (c) Escreva um algoritmo para inserir um elemento Y imediatamente após o k-ésimo elemento.

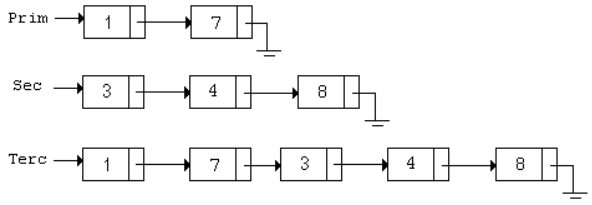
3. Seja L = (a_1, a_2, a_3, ..., a_n) uma lista linear representada num array V [1..*n*] usando o mapeamento: o i-ésimo elemento de L é armazenado em V[i]. Escreva um algoritmo para reverter a ordem dos elementos em V, isto é, o algoritmo deve transformar V tal que V[i] contenha o elemento n – i + 1 de L. O único espaço adicional disponível para seu algoritmo é aquele para variáveis simples. A entrada para seu algoritmo é V e *n*.

4. Considere a implementação estática em Pascal de duas listas encadeadas num mesmo array.

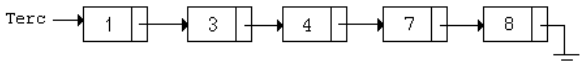
```
type        índice = 0 .. n
            nó =    record
                        info : integer;
                        lig : índice
                    end;

    lista = record
                            L : array[1..n] of nó;
                            Prim, Sec, Dispo : índice
                        end;

            var A : lista;
```

Sejam `Prim` e `Sec` "ponteiros" para o primeiro elemento de cada uma delas.

(a) Faça um programa Pascal que concatene as duas listas, Prim e Sec, dando origem a uma única lista, `Terc`. Exemplo:



(b) Faça um programa Pascal que intercale as duas listas, dando origem a uma única lista ordenada, com `Terc` apontando seu primeiro elemento. Exemplo:



Obs.: Considere as 2 listas ordenadas crescentemente pela campo chave.

5. Faça um algoritmo que inverta uma lista encadeada, isto é, o último elemento passa a ser o primeiro, o penúltimo passa a ser o segundo, e assim por diante, e o primeiro passa a ser o último. Faça a inversão através da inversão dos campos de ligação, e NÃO dos campos de informação.

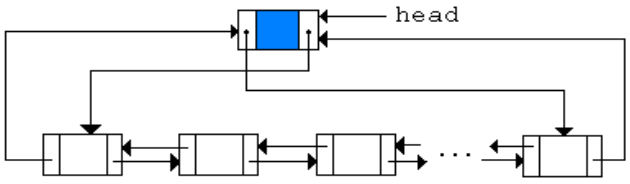
6. Faça um procedimento recursivo para imprimir uma lista encadeada.

7. Seja uma lista encadeada cujos registros possuam informação do tipo integer. Escreva um programa que ordena a lista em ordem crescente em relação à informação.

8. Faça um procedimento que remove todas as ocorrências do elemento x de uma lista encadeada.

9. Dada uma lista encadeada, faça um procedimento que conte o número de elementos desta lista.

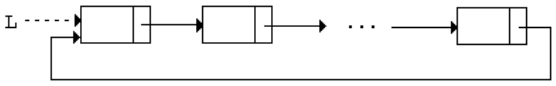
10. Uma **lista duplamente encadeada** possui registros que têm ligações com o sucessor e o predecessor na lista. Ainda, é usual ter um "nó *head*", que é um registro auxiliar que aponta para o "primeiro" e o "último" registro da lista e é apontado por eles.



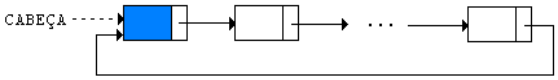
Construa um conjunto de procedimentos para busca, inserção e eliminação de elementos.

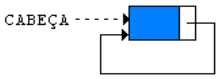
11. Escreva um procedimento para concatenar duas listas duplamente encadeadas.

12. Uma **lista encadeada circular** é uma lista encadeada cujo último elemento aponta para o primeiro:



Vantagem: cada elemento é acessível a partir de qualquer outro. Numa lista circular, não faz mais sentido se falar em primeiro ou último elemento. Porém, devemos saber, durante um percurso na lista, se já demos uma volta completa, para evitarmos *loops* infinitos. Para isso, assumimos a existência de um registro especial, chamado ***Cabeça de Lista***, cujo campo de informação não pertence ao conjunto de elementos da lista (poderia até servir de sentinela numa busca):



Situação lista circular vazia ⇒ 

Construa algoritmos para:

(a) contar o número de elementos numa lista circular;

(b) inserir um elemento à esquerda da cabeça da lista;

(c) eliminar o elemento de valor x;

(d) concatenar duas listas circulares;

(e) intercalar duas listas ordenadas;

(f) fazer uma cópia da lista;

13. Generalize a lista circular do exercício anterior para ***Lista Circular Duplamente Encadeada***, e repita os itens (a) até (f).

14. Suponha que uma lista seja declarada da seguinte forma:

```
type    nó =  record
                 bit : 0..1;
                 lig : ^ nó
              end;
lista = ^ nó;
```

Um número binário $b_1b_2...b_n$, onde cada b_i é 0 ou 1, tem o valor numérico $\sum_{i=1}^{n} bi \cdot 2^{n-i}$. Esse número binário pode ser representado pela lista b_1, b_2, ..., b_n. Essa lista, por sua vez, pode ser representada como uma lista encadeada de nós do tipo nó. Escreva um procedimento **`incremento(num_bin)`** que adiciona um ao número binário apontado por num_bin. Dica: faça incremento recursivo.