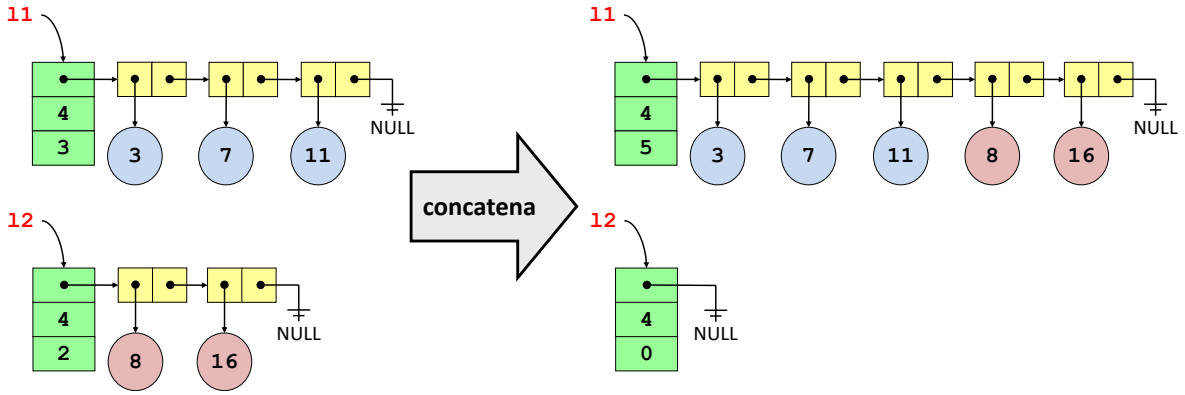
Universidade do Estado de Santa Catarina (UDESC)
Curso: Tecnologia em Análise e Desenvolvimento de Sistemas
Disciplina: Estruturas de Dados (EDA0001)
Prof. Rui J. Tramontin Jr.

Lista de Exercícios 5 – Listas Simplesmente Encadeadas

1) Faça uma função que **concatene** duas **listas encadeadas**. O resultado da concatenação deve ficar em *l1*, sendo que *l2* ficará vazia. A função deve considerar que qualquer uma das listas poderá estar vazia. Exemplo:

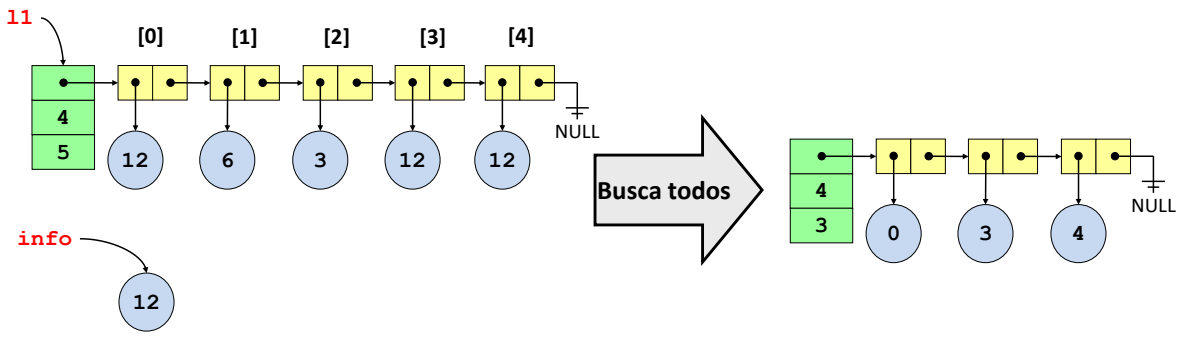


Protótipo: `void concatena( Lista *l1, Lista *l2 )`

2) Faça uma função que faça a **busca** em uma **lista encadeada**, retornando o **índice** onde a informação foi encontrada (1ª ocorrência na lista). Será preciso percorrer a lista até que se encontre o elemento cuja informação seja igual ao parâmetro *info* (usando a função de comparação). Retorna -1 caso a lista não contenha a informação. Protótipo:

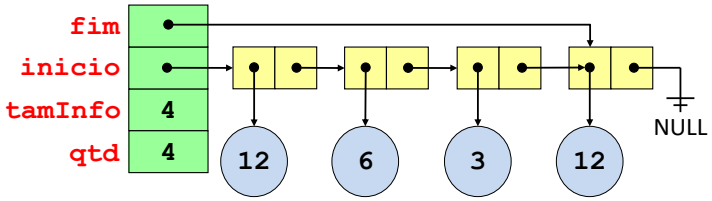
```
int busca( Lista *l, void *info, int (*compara)(void *, void *) )
```

3) Este exercício é semelhante ao anterior, mas a busca deve retornar todas as ocorrências da informação. A função recebe uma **lista encadeada**, um ponteiro para a informação e uma função de comparação (via ponteiro) e retorna uma nova *lista encadeada* contendo os **índices (posições) onde a informação se encontra**. A lista resultante conterá, portanto, valores de tipo *int*. Caso a informação não se encontre na lista original, a função deve retornar uma lista de índices vazia. Exemplo:



Protótipo: `Lista busca_todos( Lista l, void *info, int (*compara)(void*,void*) )`

4) Implemente um TAD que representa uma **Fila** usando uma **lista encadeada**. A estrutura do cabeçalho deve ser modificada levando-se em conta a referência para o *último elemento* (fim da fila). Além disso, é preciso modificar as funções de manipulação para levar em conta tal modificação. A função *inserir()* insere no fim, e a função *remover()* remove do início (*cabeça* da lista). O objetivo do ponteiro para o fim da fila (último elemento) é tornar a operação de inserção mais eficiente, pois não haverá mais a necessidade de percorrer a lista até o final. O modelo é definido conforme segue:




```
// Cabeçalho da Fila Encadeada
typedef struct {
     Elemento *inicio, *fim;
     int tamInfo, qtd;
} FilaEnc;
```

Protótipos:

```
int inserir( FilaEnc *f, void *info );
int remover( FilaEnc *f, void *info );
```

5) Escreva uma função que mostre uma **lista encadeada** na **ordem inversa**. Como o encadeamento da lista é para frente (ponteiro *proximo*), não é possível fazer o percurso na ordem inversa. Portanto, é preciso preciso utilizar alguma estratégia, conforme listado a seguir. Implemente cada uma das estratégias:

   a) Usando uma pilha: percorre-se a lista empilhando as informações; em seguida, a pilha é esvaziada e os valores são mostrados;

   b) Usando recursão (a pilha é usada implicitamente): crie uma função recursiva que percorre a lista até o último elemento; os valores são mostrados após as chamadas recursivas, ou seja, à medida que as chamadas vão retornando, os valores vão sendo mostrados (*esta solução é muito elegante, tem poucas linhas de código*);

   c) Usando estrutura de repetição (sem pilha): é preciso fazer dois laços de repetição; o laço externo repete em função da quantidade de elementos; o laço interno varia gradualmente, ou seja, a 1ª iteração vai até o último elemento, a 2ª iteração vai até o penúltimo, e assim sucessivamente.