LABORATÓRIO - EXERCICIOS

1) O que faz a função abaixo?

```
function imprimirLLE(PtLista:TipoPtNo);
begin
   while PtLista <> nil do
   begin
      write('Codigo: ');
      writeln(PtLista^.info.cod);
      write('Nome: ');
      writeln(PtLista^.info.nome);
      PtLista := PtLista^.elo;
   end;
end;
```

2) O algoritmo a seguir insere o primeiro elemento de uma Lista Duplamente Encadeada. Preencha os pontos de interrogação com as atualizações corretas para os ponteiros.

```
Proc InserirInicio (var PtLista: TipoPtnodo: Dados: TipoInfonodo);
var Pt, PtAux: TipoPtnodo;
início
   alocar (Pt);
   Pt ↑. Info := Dados;
   se PtLista = nil
   então início
            PtLista := ????;
            Pt ↑.Ant := ????;
            Pt ↑.Prox := ????;
            Fim
fim;
```

3) Construir uma lista encadeada especificando um TAD produto que contenha:

a – Um tipo de dados Produto que armazene:

i. código
ii. nome
iii. preço

b – As seguintes operações:

i. Inicializa_Lista
ii. Imprime – para listar todos os produtos

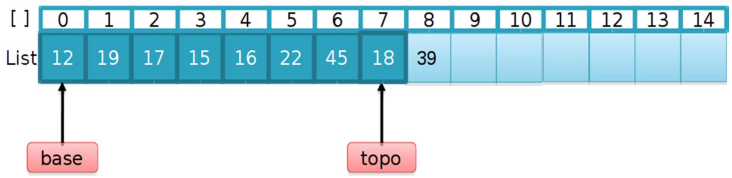
4) Implementar um programa principal que teste as operações especificadas no TAD e implemente as seguintes operações:

i. Insere_Produto – que inclua produtos em ordem crescente de nome
ii. Exclui_Produto – que exclua um produto a partir do nome informado
iii. Destrói_lista - destruir toda a lista no final do programa.

5) Avalie as complexidades das operações. Utilize a notação *O*.

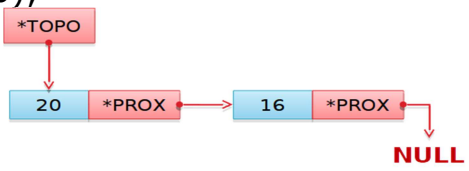
Pilha (contiguidade física):

void push(t_pilha *p, TIPO dado);
void pop(t_pilha *p);
TIPO* top(t_pilha *p);



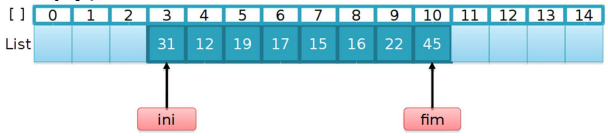
Pilha (encadeamento):

void push(t_pilha *p, TIPO dado);
void pop(t_pilha *p);
TIPO* top(t_pilha *p);



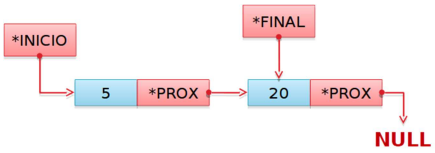
Pilha (contiguidade física):

void push(t_pilha *p, TIPO dado);
void pop(t_pilha *p);
TIPO* top(t_pilha *p);



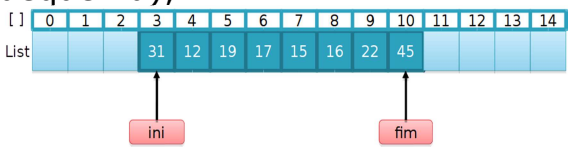
Fila (encadeamento):

void enqueue(t_fila *f, TIPO dado);
void dequeue(t_fila *f);
TIPO* front(t_fila *f);
TIPO* back(t_fila *f);



Deque (contiguidade física):

void push_front(t_deque *d, TIPO dado);
void push_back(t_deque *d, TIPO dado);
void pop_front(t_deque *d);
void pop_back(t_deque *d);



Deque (encadeamento):

void push_front(t_deque *d, TIPO dado);
void push_back(t_deque *d, TIPO dado);
void pop_front(t_deque *d);
void pop_back(t_deque *d);