Solução para a Prova – P1

Atividade avaliativa
Estruturas de Dados 1 (1001502) - ENPE – Bloco B – 05/05 a 29/06 2021

Descrição
- Proponha solução para a prova abaixo, individualmente.

O que entregar
- **Documento único, formato DOC ou PDF**, contendo nome, RA, enunciado da prova e as soluções propostas.

Quando entregar
- A prova tem duração de 2 (duas) horas. Entregue até, no máximo, 2 horas após o início da prova.

Onde entregar
- No ambiente de interação da disciplina no Google Classroom, no link indicado.

Nome:_____ RA_____ Data:_____

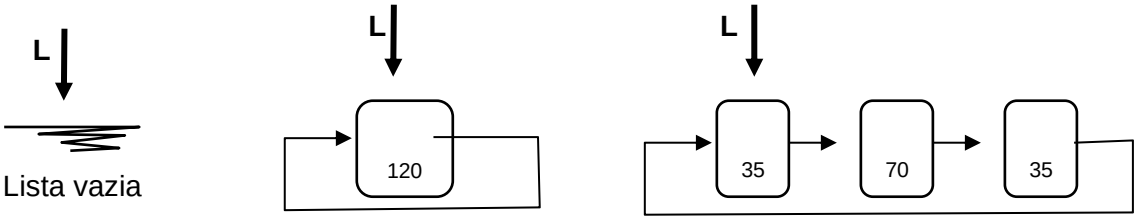
Orientações Gerais
- **Tempo para elaboração da prova: 2 horas**.

Orientações Quanto a Notação, Nomes das Variáveis, e Estruturas
- Use os **mesmos nomes fornecidos no enunciado** (L, F, X, etc.). Utilize variáveis auxiliares temporárias, o tanto quanto for necessário. É só declarar e usar. Mas **não considere a existência de nenhuma outra variável permanente**, além das definidas no enunciado. Não considere prontas para uso nenhuma operação, salvo se explicitamente indicado no enunciado da questão.
- Considere as **estruturas exatamente conforme definido no enunciado**, seja no texto da questão, seja nos diagramas.
- Para o desenvolvimento de algoritmos, use preferencialmente a notação adotada no Livro Texto: **p = NewNode; Deletenode(P), P->Info e P->Next,** sendo P uma variável do tipo **NodePtr** (ponteiro para nó). Quando a estrutura for duplamente encadeada, ao invés de P->Next considere que a notação contenha **P->Dir e P->Esq.** Também é possível implementar em C ou **C++**.

Questão 1 (4 pontos) Considere o Tipo Abstrato de Dado **Lista Cadastral** implementado através de uma **lista encadeada, circular, não ordenada, com elementos repetidos**, conforme os diagramas abaixo. **L** é um ponteiro para o início da lista. A lista vazia é composta pelo ponteiro L apontando para NULL. Implemente, **da forma mais apropriada para proporcionar portabilidade e reusabilidade**, as operações:

- **Remove-1** (variável por referência **L** do tipo NodePtr; variável **X** do tipo Inteiro, variável por referência Ok do tipo Boolean);
  /* remove uma única ocorrência do elemento de valor X da lista L. Qualquer uma das ocorrências de X. Caso nenhuma ocorrência de X for encontrada na lista L, o parâmetro Ok deve retornar FALSO. Ok deve retornar VERDADEIRO se uma ocorrência de X for encontrada e removida. Tipo NodePtr = ponteiro para Node */

- **Remove-Todos** (variável por referência **L** do tipo NodePtr; variável **X** do tipo Inteiro, variável por referência Ok do tipo Boolean);
  /* remove todas as ocorrências de valor X da lista L. Caso nenhuma ocorrência de X for encontrada na lista L, o parâmetro Ok deve retornar FALSO. Ok deve retornar VERDADEIRO se pelo menos uma ocorrência de X for encontrada e removida. Tipo NodePtr = ponteiro para Node */



1) Solução:

Remove-1 (variável por referência **L** do tipo NodePtr; variável **X** do tipo elemento, variável por referência Ok do tipo boolean) {

```
{
Se L == Null  /* caso 1: lista vazia */
Então Ok = Falso;
senão       Se (L == L->Next) and (X == L->Info)  /* caso 2: X está em L, e X é o único elemento da lista */
            Então       {  DeleteNode(L);
                        L = Null;
                        Ok = Verdadeiro;  }
            Senão       { NodePtr PAux, Ant;     /* Casos 3 e 4: lista tem vários elementos; X no meio/final da lista (caso 3) ou em L (caso 4) */
                        Ant = L;
                        PAUx = L->Next;
                        Enquanto (PAux->Info != X) AND (PAux != L) Faça {
                                    Ant = PAux;
                                    PAux = PAux -> Next; }
                        Se PAux->Info == X
                        Então       { Ant->Next = PAux ->Next;
                                    Se PAux == L Então { L=L->Next;}   /* caso 4 - X está em L; lista tem vários elementos; L precisa avançar */
                                    DeleteNode(PAux);
                                    Ok = Verdadeiro; }
                        senão       Ok = Falso;
}
```

Comentários sobre a correção da questão 1-a: Dos 4 pontos da questão, Remove-1 vale 3 pontos. No Remove_1 é preciso tratar todos os casos. **Caso 1**: lista vazia; **caso 2**: remove o único elemento da lista. **Caso 3**: lista tem vários elementos e X está no meio da lista; **caso 4**: a lista tem vários elementos, e X é o primeiro da lista. Se um caso não for tratado, ou se for tratado de modo inadequado, será descontado de 1 ponto. É comum o(a) estudante esquecer de tratar o Caso 4. Se a solução tiver erro na condição de repetição, fazendo-a entrar em loop ou algo assim, será descontado em torno de 1 ponto. Em geral, a principal dica dada aos que cometem erros nesse tipo de exercício é: praticar! Podem ser úteis também as orientações quanto ao modo de projetar as soluções: identificar todos os casos, desenhar cada caso, tratá-los separadamente, fazer o algoritmo de modo simples, testar o algoritmo passo a passo, desenhando.

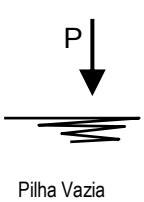
Remove-Todos (variável por referência **L** do tipo NodePtr; variável **X** do tipo elemento, variável por referência Ok do tipo boolean) {

```
{
variável Removeu-pelo-menos-1 do tipo Boolean;

Removeu-pelo-menos-1 = Falso;

Repita
    Remove-1(L, X, Ok);
    Se (Ok == Verdadeiro) Então { Removeu-pelo-menos-1 = Verdadeiro;}
Até que Ok = Falso;
Ok = Removeu-pelo-menos-1;
}
```

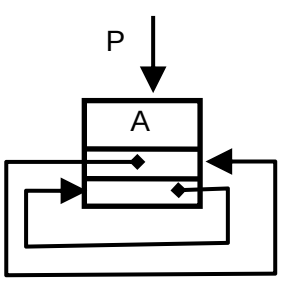
Comentários sobre a correção da questão 1-b: Dos 4 pontos da questão, o Remove-Todos vale 1 ponto. O enunciado da questão 1 diz "Implemente, da forma mais apropriada para proporcionar portabilidade e reusabilidade". Assim, o ponto chave no Remove-Todos é não "abrir a TV". Deve ser implementado através de chamadas ao Remove_1, sem qualquer referência à estrutura de armazenamento. A implementação fica muito fácil. Se abrir a TV, descontar o 1 ponto da questão.

Questão 2 (3 pontos) Considere o Tipo Abstrato de Dados **Pilha** implementado através de uma **lista duplamente encadeada, circular,** segundo os diagramas abaixo. No diagrama à esquerda, a Pilha P está vazia. No diagrama ao centro a Pilha P contém um único elemento, e no diagrama mais à direita, o elemento de valor **D está no topo** (ordem de entrada A, B, C e D). Implemente, da forma mais apropriada para proporcionar portabilidade e reusabilidade, as operações:

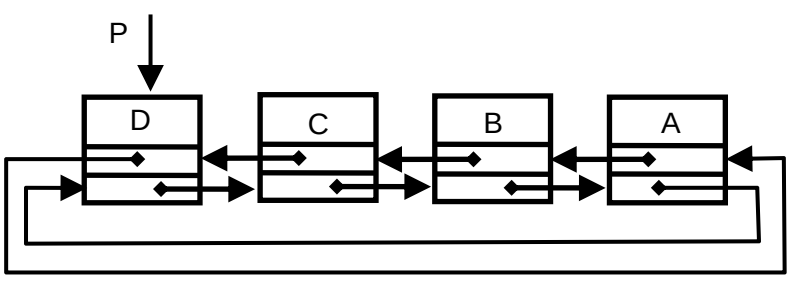
- **Empilha** (variável por referência P do tipo NodePtr, variável X do tipo Char);
  /* empilha o elemento X na Pilha P. Desconsidere a possibilidade de a pilha estar cheia. Tipo NodePtr = ponteiro para nó */



Pilha Vazia



Pilha com 1 elemento



Pilha com 4 elementos, e D está no topo (ordem de entrada: A, B, C e D)

2- Solução:

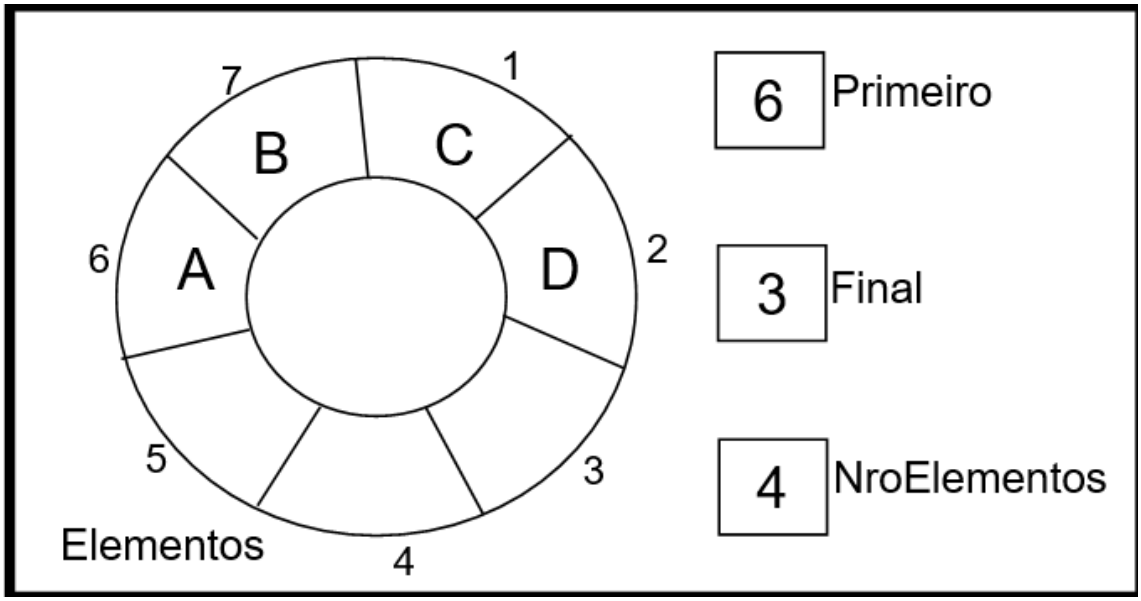
```
Empilha (parâmetro por referência P do tipo Pilha, parâmetro X do tipo Char)
{
Variável PAux do tipo NodePtr; // ponteiro para Nó, auxiliar

Paux = NewNode;
PAux->Info = X;
Se (P = Null)
Então       { // caso 1: a Pilha P inicialmente está vazia e receberá o primeiro elemento, que passará a ser apontado por P
            PAux->Dir = Paux;
            PAux->Esq = PAux;
            P = PAux;
            PAux = Null;
            }
Senão       { // caso 2: a pilha P já contém elementos; deve ser inserido à esquerda de P, e P passa a apontar para o novo elemento
            PAux-> Dir = P;
            PAux->Esq  = P->Esq;
            P->Esq->Dir = PAux;
            P->Esq = PAux;
            P = PAux;
            PAux = Null;
            }
} // fim do Empilha
```

Comentários sobre a correção da questão 2: Neste exercício, Empilha, os erros mais comuns surgem da dificuldade de "visualizar" o que está acontecendo no desenho. O(a) estudante pode, por exemplo, deixar de ajustar adequadamente as ligações da lista. O(a) estudante pode ainda deixar de tratar um dos casos (pilha vazia / não vazia). Nesses erros, em geral, a principal orientação é "desenhar passo a passo a modificação da estrutura a partir de seu próprio algoritmo". Fazendo isso, é mais fácil enxergar o que está errado. Será descontado em torno de 1 ponto, dependendo do erro.

Questão 3 (3 pontos) Considere o Tipo Abstrato de Dados **Fila** implementado com **alocação sequencial e estática**, com um **vetor circular,** sem realocação de elementos, segundo o diagrama abaixo. Implemente, da forma mais apropriada para proporcionar portabilidade e reusabilidade, a operação:

Insere (parâmetro por referência F do tipo Fila, parâmetro X do tipo Char, parâmetro por referência DeuCerto do tipo Boolean) {

/* Insere o elemento X na Fila F. O parâmetro DeuCerto deve indicar se a operação foi bem sucedida ou não. A operação só não será bem sucedida se tentarmos inserir um elemento em uma Fila cheia. Uma Fila F do tipo Fila é composta por 4 campos: o vetor F.Elementos, e os inteiros F.Primeiro, F.Final e F.NroDeElementos. F.Primeiro indica a posição do primeiro da Fila, F.Final indica a primeira posição vaga do vetor, e F.NroDeElementos indica a quantidade de elementos na fila F, em determinado momento. Considere que a constante TamanhoDoVetor, que indica o número de elementos que podem ser armazenados da Fila. Os elementos da Fila são do tipo Char. Considere ainda pronta para uso a operação Cheia(F), que verifica se uma fia F está cheia ou não. Ou seja, é possível utilizar, e não é necessário desenvolver a operação Cheia(F). */



Solução:

Insere (parâmetro por referência F do tipo Fila, parâmetro X do tipo Char, parâmetro por referência DeuCerto do tipo Boolean)

```
{
Se (Cheia(F)== Verdadeiro)
Então DeuCerto = Falso;                          // ... não podemos inserir
Senão { DeuCerto = Verdadeiro;                   // inserindo X na Fila F... operação deu certo..
        F.NroElementos = F. NroElementos + 1; // aumenta o número de elementos
        F.Elementos[ F.Final ] = X;              // insere X no final da Fila F

                                        // avançando o apontador Final... Atenção: o vetor é circular
        Se (F.Final == Tamanho)    // se Final estiver na última posição do vetor..
        Então F.Final = 1;         // ..avança para a primeira posição (ajustar para 0?)
        Senão F.Final = F.Final + 1;   // senão avança para a próxima posição
       }; // fim do senão
} // fim da operação Insere
```

Comentários sobre a correção de toda a prova:

- Outros erros que podem surgir: alocar um nó quando o que é necessário é alocar apenas um ponteiro; utilizar elementos não definidos no enunciado (exemplo, tamanho da lista/pilha); implementar com técnica de alocação diferente da solicitada (sequencial/estática, encadeada/dinâmica); trocar nomes dos parâmetros propostos; desenvolver algo diferente do previsto no enunciado. Será descontado da nota um valor proporcional ao erro cometido.