Prova 1 - Solução

Atividade Avaliativa
Estruturas de Dados 1 (1001502) - ENPE 4 – 2022

Orientações Gerais

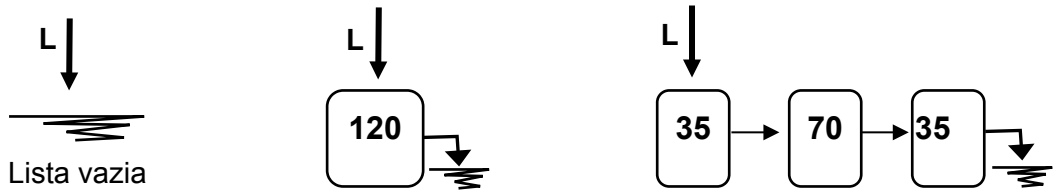
- **Tempo para elaboração da prova: 3h**.

Orientações Quanto a Notação, Nomes das Variáveis, e Estruturas

- Use os **mesmos nomes fornecidos no enunciado** (L, X, P, P.Header, L1, L2, 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++**. Nesse caso usar mesma nomenclatura e tipos equivalentes ao que foi fornecido no enunciado. Os tipos **booleanos em C** podem ser substituídos para **inteiros,** com 0 equivalente a FALSO e 1 equivalente a VERDADEIRO.

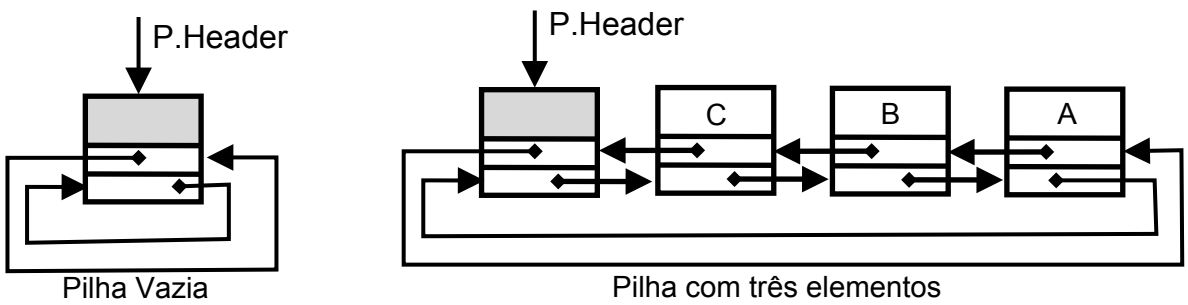
Questão 1 (4 pontos). Considere o Tipo Abstrato de Dado **Lista Cadastral** implementado através de uma **lista encadeada, não ordenada, com elementos repetidos**, conforme os diagramas abaixo. **L** é um ponteiro para o início da lista. A lista vazia é composta pelo ponteiro L apontando para NULL. Implemente,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



Questão 2. (3 pontos) Considere o Tipo Abstrato de Dados Pilha implementado através de uma **lista duplamente encadeada, circular, com nó Header,** segundo os diagramas abaixo. No diagrama a esquerda, a Pilha P está vazia. No diagrama a direita, a pilha contém 3 elementos – A, B e C. Se seu primeiro nome começa com as letras A, B, .. M, considere que A está no topo e a sequência de entrada na pilha foi C, B, A. Se seu primeiro nome começa com as letras N, O, ... Z, considere que C está no topo e a sequência de entrada na pilha foi A, B, C. Implemente as operações:

- **Boolean Vazia**(variável por referência P do tipo Pilha) **//** retorna True se a Pilha não tiver nenhum elemento; falso caso contrário.
- **Empilha** (variável por referência P do tipo Pilha, variável X do tipo Elemento) /* insere 1 elemento na Pilha P. Desconsidere a possibilidade de a Pilha estar cheia */



Questão 3 (2 pontos) Em determinada localidade temos a Lista de Pessoas Vacinadas (L1) e temos também a Lista de Pessoas Já Infectadas por determinado vírus (L2). L1 e L2 não possuem repetições, ou seja, em L1 há um único registro para cada pessoa, ainda que tenha tomado várias doses, e em L2 há um único registro para cada pessoa, ainda que tenha sido infectada várias vezes. Mas é possível que uma mesma pessoa já tenha sido vacinada e (também) infectada; caso em que o registro da pessoa apareceria tanto em L1 quanto em L2.

Desenvolva uma função que receba as listas L1 e L2 e retorne o número de pessoas que já foram ou vacinadas, ou infectadas, ou ambos.

Int NroPessoasVacinadasOuInfectadas(Lista L1, Lista L2);
// Calcula e retorna o número de pessoas que foram vacinadas (L1), ou infectadas (L2) ou ambos.
// Considere que o elemento das listas seja do tipo "Pessoa".

Desenvolva fazendo uso das seguintes operações de uma Lista Cadastral:

- Cria(L); // Cria uma Lista Cadastral L, iniciando sua situação como vazia;
- Vazia(L); // Verifica se a Lista Cadastral L está vazia, retornando true para vazia, e false caso contrário;
- Insere (L,X,Ok); // Insere o elemento de valor X na Lista L.O parâmetro Ok deve retornar true se a operação foi bem sucedida; false caso não;
- Retira(L,X,Ok); // Retira da Lista L o elemento de valor X, caso X estiver na Lista. Neste caso, o parâmetro Ok deve retornar true; false caso contrário;
- Boolean EstaNaLista (L, X); // Verifica se o elemento de valor X faz parte da Lista Cadastral L, retornando true caso X estiver na Lista L, e false caso não;
- PegaOPrimeiro(L, X, TemElemento); // X retorna o valor do primeiro elemento da Lista L, se esse primeiro elemento existir. Se não existir esse primeiro elemento (Lista vazia), o parâmetro TemElemento retornará false. Se existir, retornará true;
- PegaOPróximo(L,X,TemElemento); // X retorna o valor do próximo elemento da Lista, em relação à última chamada à uma das operações PegaOPrimeiro ou PegaOPróximo. Se não existir esse próximo elemento (final da Lista), o parâmetro TemElemento retornará o valor Falso. Se existir, retornará true.