**Lista de Exercícios TAD Lista - 2ª Etapa**

**Deve ser resolvida em dupla**

*Entregar manuscrita no dia da prova. Lista digitalizada vale 50% da nota.*

1. Qual a diferença entre alocação estática de memória e alocação dinâmica de memória?
2. Qual o valor de **y** no final do programa? Tente primeiro descobrir e depois verifique no computador o resultado. A seguir, escreva um comentário em cada comando de atribuição explicando o que ele faz e o valor da variável à esquerda da atribuição após a execução.

int main(){

int y, \*p, x;

y = 0;

p = &y;

x = \*p;

x = 4;

(\*p)++;

x--;

(\*p) += x;

printf(“y = %d\n”,y);

}

1. Quais as vantagens e desvantagens de uma lista implementada com vetor (arranjo) para uma lista implementada com ponteiro (encadeada).
2. Utilizando o TAD lista encadeada, modele e implemente um sistema de cadastro de alunos de uma universidade com as informações: nome, curso, período e turno. Além das estruturas necessárias para tal implementação crie a **assinatura** das funções para: criarListaAlunosVazia, cadastrarAluno, removerAluno e pesquisarAluno.
3. Uma Loja de informática deseja desenvolver um sistema computacional para controlar seu estoque de produtos. Para ajudar esta empresa devemos propor as ESTRUTURAS para uma LISTA ENCADEADA contendo as seguintes características: A estrutura da LISTA deve armazenar informações sobre os produtos, contendo: Nome do Produto, Quantidade em Estoque, Valor Unitário, Data da Compra (a data deve ser uma estrutura separada, com Dia, Mês e Ano) e Nome do Fornecedor. Crie a assinatura de 4 funções (use a sua criatividade) para manipular tais estruturas.

**Obs:** Nos exercícios 4 e 5 não precisa implementar as funções, coloque somente as assinaturas com os respectivos parâmetros.

1. Complemente o código de Lista Encadeada com as seguintes funções:
   1. Dadas duas listas encadeadas L1 e L2, implementar a função **UNION**, que coloque em uma lista encadeada L3 a união entre as duas listas.
   2. Dadas duas listas encadeadas L1 e L2, sem elementos repetidos, implemente a função **INTER**, que coloque em uma lista L3 encadeada a intersecção entre as duas listas, também sem elementos repetidos.
   3. Dadas duas listas encadeadas (L1 e L2) ordenadas, implemente a função **MERGE**, que coloca em uma lista L3 encadeada, a junção das duas listas de forma que L3 também fique ordenada.
   4. Implemente uma função (**pesquisaElemento**) para verificar se um elemento X está na lista L encadeada.
   5. Seja uma lista L encadeada e um inteiro N, implemente uma função (**pegaElementoPosicao**) para pegar o valor do n-ésimo elemento da lista L. Não é para retirar o elemento da lista.
   6. Implemente uma função (**inverteLista**) que inverta os elementos de uma lista encadeada.
   7. Implemente uma função (**calculaTamanhoLista**) para calcular o tamanho de uma lista encadeada.
   8. Implemente uma função (**verifica2Listas**) para verificar se duas listas encadeadas L1 e L2 são iguais.
   9. Implemente uma função (**insereOrdenado**) que deve inserir o elemento sempre mantendo os números da nova lista ordenados. Ao identificar a posição a ser inserida faça uso das funções implementadas insereInicio, insereMeio ou insereFinal.
   10. Implementar uma função que insere um elemento do meio da lista.

void insereMeio (Lista \*lista, Item x, int pos);

* 1. Implementar uma função que retira um elemento do meio da lista.

void retiraMeio (Lista \*lista, Item x, int pos);

**Importante:** o valor do parâmetro **pos** deve ser uma posição entre o primeiro e o último elemento da lista.