UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Faculdade da Computação
1º Trabalho de Algoritmos e Estrutura de Dados 1
Prof. Luiz Gustavo Almeida Martins

- ✓ Deve ser entregue um arquivo zip com os códigos das questões organizados em pastas (uma pasta por questão/TAD), sendo a integridade desse arquivo de responsabilidade dos alunos.
- ✓ A apresentação individual deve ser agendada com o professor para a semana especificada no plano de ensino.
- ✓ Os códigos deverão ser implementados somente em Linguagem C, sendo necessária a utilização das estruturas de dados conforme discutidas em sala.
- ✓ Deve-se aproveitar o conhecimento da estrutura e seu funcionamento para buscar a maior eficiência da lógica adotada na implementação das operações de um TAD.
- ✓ Em todas as questões, além das operações vistas nas aulas (ex: *cria_lista, lista_vazia, lista_cheia, insere, remove, esvazia_lista, apaga_lista, get_elem_pos*), deve-se implementar um programa aplicativo que permita executar repetidamente as operações do TAD.
- ✓ **Antes de cada operação, deve-se colocar um comentário com a sua especificação.**

1) Implementar o TAD **lista ordenada** decrescente de números inteiros com no máximo 20 elementos, usando alocação **estática/sequencial**. Além das operações vistas em sala, o TAD também deve contemplar:
   - Remover negativos: remove todos os elementos negativos da lista.
   - Remover pares: remove todos os elementos pares da lista.
   - Tamanho: retorna o número de elementos da lista
   - Intercalar: recebe duas listas ordenadas (L1 e L2) e retorna uma nova lista L3 formada pelos elementos de L1 e L2 intercalados, mantendo o critério de ordenação. As listas originais não devem ser alteradas.

2) Implementar o TAD **lista não ordenada** de strings com no máximo 10 elementos, cada um com até 15 caracteres, usando alocação **estática/sequencial**. Além das operações vistas em sala, o TAD também deve contemplar:
   - Remover todas: remove todas as strings da lista que começam com um dado caractere.
   - Remover maior: remove e retorna a maior string da lista. No caso de empate, deve-se remover a primeira ocorrência encontrada.
   - Tamanho: retorna o número de elementos da lista.
   - Concatenar: recebe duas listas (L1 e L2) e retorna uma nova lista L3 com os elementos de L1 seguidos dos elementos de L2. As listas originais não devem ser alteradas.

3) Implementar o TAD **lista não ordenada** de números reais (*float*) usando alocação **dinâmica/encadeada SEM cabeçalho**. Além das operações vistas em sala, o TAD também deve contemplar:
   - Tamanho: retorna o número de elementos da lista.
   - Remover menor: remove e retorna o menor número da lista. No caso de empate, deve-se remover a última ocorrência encontrada.
   - Iguais: recebe duas listas ordenadas e verifica se elas são iguais.
   - Intercalar: recebe duas listas ordenadas e retorna uma nova lista com os elementos das duas listas de entrada intercalados. As listas originais não devem ser alteradas.

4) Implementar o TAD **lista ordenada** crescente de números inteiros usando alocação **dinâmica/ encadeada COM cabeçalho**. Além das operações vistas em sala, o TAD também deve contemplar:
   - Tamanho: retorna o número de elementos da lista
   - Remover elemento da posição: remove o elemento que se encontra na posição indicada, retornando seu valor. Se a posição não existir na lista, a operação deve indicar falha.
   - Inverter: recebe uma lista L e retorna uma nova lista L2, formada pelos elementos de L na ordem inversa. A lista original não deve ser alterada.
   - Intercalar: recebe duas listas ordenadas e retorna uma nova lista com os elementos das duas listas de entrada intercalados, mantendo o critério de ordenação. As listas originais não devem ser alteradas.

5) Implementar o TAD **lista não ordenada** de caracteres usando alocação **dinâmica com encadeamento CÍCLICO**. Além das operações vistas em sala, o TAD também deve contemplar:
   - Inserir no início: inserir o elemento no início da lista
   - Inserir na posição: insira o elemento em uma posição indicada. A operação deve verificar se a posição desejada é válida.
   - Remover no fim: remover o último elemento da lista, retornando seu valor para a aplicação.
   - Remover vogais: remove todos os elementos da lista que são vogais.

6) Implementar o TAD **lista não ordenada** de números reais (*double*) usando alocação **dinâmica com encadeamento duplo**. Além das operações vistas em sala, o TAD deve contemplar:
   - Remover todos: remove todas as ocorrências de um elemento da lista
   - Remover maior: remove o maior elemento encontrado na lista, retornando seu valor. No caso de empate, deve-se remover a primeira ocorrência encontrada.
   - Inserir na posição: insira o elemento em uma posição indicada. A operação deve verificar se a posição desejada é válida.
   - Inverter: recebe uma lista L e retorna uma nova lista L2, formada pelos elementos de L na ordem inversa. A lista original não deve ser alterada.