UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Faculdade da Computação
Primeiro Trabalho Individual de AED1 – Valor 8 Pontos
Profa. Gina Maira B. de Oliveira

- Deve ser enviado até **02/10/2019** para o estagiário Thiago Fialho (exceto exercício 8). Email: **fialhot@gmail.com**
- A apresentação individual dos códigos será agendada posteriormente.
- Os códigos deverão ser implementados somente em Linguagem C, sendo necessária a utilização das estruturas de dados conforme discutidas em sala.

1) Fundamentos de C (vetores, structs, ponteiros e alocação dinâmica):

a) Construa um programa que manipula um vetor de structs chamado **contratos** (alocado dinamicamente), sendo que a struct chamada **dados** possui um campo nome (apenas primeiro nome) e outro campo montante que armazena um inteiro. O número de elementos do vetor deve ser informado pelo usuário. Os valores do montante e nome de cada estrutura devem ser digitados pelo usuário no próprio programa main (). Após a digitação, imprimir o vetor de entrada.Posteriormente, o programa manipula esses dados de forma que todo nome dentro de uma struct deve ser deve ter suas vogais eliminadas (ex: se o nome for "Camila", deve ficar "Cml"). O campo montante permanece inalterada. Ao final, imprimir o vetor resultante, após a digitação do usuário e o processamento das vogais.
Obs1: A alocação das estruturas e a leitura dos dados deve ser implementado na própria main(). Ao final, liberar o espaço do vetor.
Obs 2: Implementar a função percorre_contratos() que manipula o vetor percorrendo todas as structs e a função altera_nome() que manipula uma única string (essa deve receber a struct correspondente, e não apenas o campo "nome").

b) Repetir o exercício anterior, porém a alocação das estruturas e a leitura dos dados devem ser implementados em uma função auxiliar aloca_structs(). Obs: deve ser usado ponteiro duplo nessa função.

2) Implementar o TAD **lista não ordenada** usando alocação **estática/seqüencial**. A TAD deve conter todas as operações vistas em sala e laboratório: **cria_lista, lista_vazia, lista_cheia, insere_elem, remove_elem, imprime_lista** e **remove_todos**, além de incorporar as operações a seguir:

- **Inserir no início:** inserir o elemento no início da lista (obs: a insere_elem vista em sala insere no final da lista, para essa forma de implementação).
- **Remover ímpares**: remove todos os elementos ímpares da lista.
- **Menor**: retorna o menor elemento da lista
- **Tamanho**: retorna o número de elementos da lista
- **Concatena**: recebe duas listas (L1 e L2) e retorna uma nova lista L3 com os elementos de L1 seguidos dos elementos de L2. OBS: a solução deve ser pensada de forma eficiente e cada lista (L1, L2 e L3) deve ser prcorrida uma única vez durante a construção de L3. Dessa forma, funções do tipo "insere" não devem ser utilizadas. O mesmo é valido para outras TADs, na implementação do **Concatena**.

3) Implementar o TAD **lista ordenada** usando alocação **estática/seqüencial**. A TAD deve conter todas as operações vistas em sala e laboratório: **cria_lista, lista_vazia, lista_cheia, insere_ord, remove_ord, imprime_lista** e **intercala**, além de incorporar as operações a seguir:

- **Remover pares**: remove todos os elementos pares da lista.
- **Maior**: retorna o maior elemento da lista
- **Tamanho**: retorna o número de elementos da lista
- **Iguais**: recebe duas listas ordenadas e verifica se elas são iguais
- **Intercala**: recebe duas listas ordenadas (L1 e L2) e retorna uma nova lista L3 com os elementos de L1 e L2 intercalados conforme a ordenação. OBS: a solução deve ser pensada de forma eficiente e cada lista (L1, L2 e L3) deve ser prcorrida uma única vez durante a construção de L3. Dessa forma, funções do tipo "insere" não devem ser utilizadas. O mesmo é valido para outras TADs, na implementação do **Intercala**.

4) Implementar o TAD **lista não ordenada** usando alocação **dinâmica/encadeada simples (SEM cabeçalho)**. A TAD deve conter todas as operações vistas em sala e laboratório: **cria_lista, lista_vazia, lista_cheia, insere_elem, remove_elem, imprime_lista** e **remove_todos**, além de incorporar as operações a seguir:

- **Inserir no final:** inserir o elemento no final da lista (obs: a insere_elem vista em sala insere no início da lista, para essa forma de implementação).
- **Remover ímpares**: remove todos os elementos ímpares da lista.
- **Menor**: retorna o menor elemento da lista
- **Tamanho**: retorna o número de elementos da lista
- **Concatena**: recebe duas listas (L1 e L2) e retorna uma nova lista L3 com os elementos de L1 seguidos dos elementos de L2. Considerar a eficiência do código.

5) Implementar o TAD **lista ordenada** usando alocação **dinâmica/encadeada simples (SEM cabeçalho)**. A TAD deve conter todas as operações vistas em sala e laboratório: **cria_lista, lista_vazia, lista_cheia, insere_ord, remove_ord, imprime_lista** e **intercala**, além de incorporar as operações a seguir:

- **Remover pares**: remove todos os elementos pares da lista.
- **Menor**: retorna o menor elemento da lista
- **Maior**: retorna o maior elemento da lista
- **Tamanho**: retorna o número de elementos da lista
- **Iguais**: recebe duas listas ordenadas e verifica se elas são iguais

6) Implementar o TAD **lista ordenada** usando alocação **dinâmica/encadeada COM cabeçalho**. A TAD deve conter todas as operações vistas em sala e laboratório: **cria_lista, lista_vazia, lista_cheia, insere_ord, remove_ord, imprime_lista** e **tamanho**, além de incorporar as operações a seguir:

- ~~**Remover pares**: remove todos os elementos pares da lista.~~
- **Menor**: retorna o menor elemento da lista
- **Iguais**: recebe duas listas ordenadas e verifica se elas são iguais.

- **Intercalar**: recebe duas listas ordenadas (L1 e L2) e retorna uma nova lista L3 com os elementos de L1 e L2 intercalados conforme a ordenação. Considerar a eficiência do código.

7) Implementar o TAD **lista não ordenada** usando alocação **dinâmica com encadeamento CÍCLICO**. A TAD deve conter todas as operações vistas em sala e laboratório: **cria_lista, lista_vazia, lista_cheia, insere_elem, remove_elem, imprime_lista** e **tamanho**, além de incorporar as operações a seguir:

- **Inserir no início:** inserir o elemento no início da lista
- **Inserir na posição:** insira o elemento em uma posição definida na chamada da função. A operação deve verificar se a posição desejada é valida.
- **Remove elemento da posição:** remover o elemento que se encontra na posição definida na chamada da função. Se a posição não existir na lista, a operação deve indicar falha.
- **Maior**: retorna o maior elemento da lista.

8) (*A entrega desse exercício não será obrigatória na data de 01/10. Ele fará parte da 2ª lista de exercícios).
Implementar o TAD **lista não ordenada** usando alocação **dinâmica com encadeamento duplo**. A TAD deve conter todas as operações vistas em sala e laboratório: **cria_lista, lista_vazia, lista_cheia, insere_elem, remove_elem, imprime_lista** e **tamanho**, além de incorporar as operações a seguir:

- **Remover todos**: remove todas as ocorrências de um elemento da lista
- **Remover maior**: remove o maior elemento encontrado na lista.
- **Multiplos de 3:** retornar uma lista L2 formada pelos elementos da lista de entrada L, que são múltiplos de 3.