UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Faculdade da Computação
Trabalho Individual de AED1 – Valor 10 Pontos
Profa. Gina Maira B. de Oliveira

- Deve ser entregue até **29/04/2019**
- 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) Especificar e implementar (através de projeto de TAD) o TAD **Ponto**, para representar pontos no espaço bidimensional (x,y). Este TAD deve permitir criar de um ponto, liberar a memória alocada para um ponto e calcular a distância entre dois pontos. O programa principal (aplicativo) deve ler as coordenadas de 4 pontos (digitadas pelo usuário). Posteriormente, deve calcular e imprimir na tela a distância entre cada par de pontos. Para isto, este aplicativo deverá: a) criar os quatro pontos, b) calcular e mostrar a distância entre eles, c) definir e mostrar as menores e maiores distâncias. Ao final, deve liberar a memória alocada para os pontos.

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, tamanho** e **menor**, 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 todos:** recebe um valor inteiro (Elem) e deve remover todas as ocorrências de Elem em uma lista.
   - **Remover ímpares**: remove todos os elementos ímpares 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 funções que percorrem uma ou mais listas, devem percorrer cada lista uma única vez.

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**, **tamanho** e **menor,** além de incorporar as operações a seguir:
   - **Remover pares**: remove todos os elementos pares da lista.
   - **Intercala:** essa função recebe duas listas ordenadas de inteiros (que podem ter tamanhos diferentes) e deve elaborar uma nova lista ordenada que seja a junção dos elementos das duas listas.
   - **Iguais**: recebe duas listas ordenadas e verifica se elas são iguais

* As funções que percorrem uma ou mais listas, devem percorrer cada lista uma única vez.

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, tamanho** e **menor**, 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 todos:** recebe um valor inteiro (Elem) e deve remover todas as ocorrências de Elem em uma lista.
- **Remover pares**: remove todos os elementos ímpares 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 funções que percorrem uma ou mais listas, devem percorrer cada lista uma única vez.

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, tamanho** e **maior**, além de incorporar as operações a seguir:
   - **Remover ímpares**: remove todos os elementos pares da lista.
   - **Intercala:** essa função recebe duas listas ordenadas de inteiros (que podem ter tamanhos diferentes) e deve elaborar uma nova lista ordenada que seja a junção dos elementos das duas listas.
   - **Iguais**: recebe duas listas ordenadas e verifica se elas são iguais

   * As funções que percorrem uma ou mais listas, devem percorrer cada lista uma única vez.

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**, **tamanho e maior**, além de incorporar as operações a seguir:
   - **Remover ímpares**: remove todos os elementos pares 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.

   * As funções que percorrem uma ou mais listas, devem percorrer cada lista uma única vez.