



UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Faculdade de Computação

Avenida João Naves de Ávila, 2121, Bloco 1B - Bairro Santa Mônica, Uberlândia/MG, CEP 38400-902
Telefone: +55 (34) 3239-4218 - www.facom.ufu.br - cocom@ufu.br



Bacharelado em Ciência da Computação

Bacharelado em Sistemas de Informação

Disciplina: Algoritmos e Estruturas de Dados 1 – AED1 [GBS024/GSI006]

Prof. Me. Claudiney R. Tinoco

Material baseado: Prof. Dr. Luiz Gustavo Almeida Martins

1º Trabalho de Algoritmos e Estruturas de Dados 1

- Os trabalhos poderão ser feitos em duplas;
 - Deverá ser submetido no repositório do GitHub até o dia 06/07/2022;
 - A apresentação dos códigos será agendada pelo professor no final da disciplina;
 - Os códigos deverão ser implementados somente em Linguagem C, sendo necessária a utilização das estruturas de dados conforme discutidas em sala;
 - Cada TAD deverá ser implementando em um projeto diferente;
 - O programa cliente deverá possuir um menu interativo para executar as operações.
- 1) Implementar o TAD lista não ordenada usando alocação estática/sequencial.
Operações que o TAD deve contemplar:
 - Inicializar lista (vista em sala);
 - Verificar lista vazia (vista em sala);
 - Inserir elemento (vista em sala);
 - Remover elemento (vista em sala - Manter os elementos na ordem de entrada);
 - Remover ímpares: remove todas as ocorrências dos elementos ímpares da lista de entrada (Manter os elementos na ordem de entrada);
 - Menor: retorna o menor elemento da lista;
 - Ordena lista: ordene, em ordem crescente, os elementos da lista (entrada);
 - Tamanho: retorna o número de elementos da lista;
 - Concatena: recebe duas listas não ordenadas L1 e L2 e retorna uma nova lista com os com os elementos de L1 seguidos dos elementos de L2.
 - 2) Implementar o TAD lista ordenada usando alocação estática/sequencial. Operações que o TAD deve contemplar: as mesmas definidas no item anterior.
 - 3) Implementar o TAD lista ordenada usando alocação dinâmica/encadeada SEM cabeçalho.
Operações que o TAD deve contemplar:
 - Inicializar lista (vista em sala);
 - Verificar lista vazia (vista em sala);
 - Inserir elemento (vista em sala);
 - Remover elemento (vista em sala);
 - Tamanho: retorna o número de elementos da lista;
 - Média: retorna a média aritmética simples dos elementos da lista;
 - Iguais: recebe duas listas ordenadas e verifica se elas são iguais;
 - Intercalar: recebe duas listas ordenadas e retorna a lista com os elementos das duas listas intercalados conforme a ordenação;

- Inverter: recebe uma lista L e retorna uma nova lista L2, formada pelos elementos de L na ordem inversa;
 - Retorna ímpares: recebe uma lista L e retorna uma nova lista L2, formada apenas com os elementos ímpares de L.
- 4) Implementar o TAD lista ordenada usando alocação dinâmica/encadeada COM cabeçalho. Operações que o TAD deve contemplar: as mesmas definidas no item 3.
- 5) Implementar o TAD lista não ordenada usando alocação dinâmica com encadeamento CÍCLICO. Operações que o TAD deve contemplar:
- Inicializar lista (vista em sala);
 - Verificar lista vazia (vista em sala);
 - Inserir no final (vista em sala);
 - Remover no início (vista em sala);
 - Inserir elemento: inserir o elemento no início;
 - Inserir em posição: insira o elemento de entrada em uma posição definida pela na entrada. Verifique se esta posição é válida;
 - Remove elemento em posição: remover o elemento que se encontra na posição especificada na entrada. Se a posição não existir na lista, retorne mensagem de falha na tela;
 - Tamanho: retorna o número de elementos da lista;
 - Maior: retorna o maior elemento da lista;
 - Remover pares: remover da lista todos os seus elementos pares.
- 6) Implementar o TAD lista ordenada usando alocação dinâmica com encadeamento duplo. Operações que o TAD deve contemplar:
- Inicializar lista;
 - Verificar lista vazia;
 - Inserir elemento;
 - Remover elemento;
 - Tamanho: retorna o número de elementos da lista;
 - Média: retornar a média aritmética simples dos elementos da lista;
 - Iguais: recebe duas listas ordenadas e verifica se elas são iguais;
 - Remover todos: remove todas as ocorrências de um elemento em uma lista;
 - Remover maior: remove o maior elemento da lista encontrado. Remova todas as ocorrências deste elemento;
 - Múltiplos de 3: retornar uma lista L2 formada pelos elementos da lista de entrada L, que são múltiplos de 3.
- 7) Implementar o problema de Josephus utilizando o TAD lista.
- Problema:** um grupo de soldados está cercado pelo inimigo e existe apenas um cavalo para a fuga. Decidiu-se que o soldado que se salvará será definido na sorte, independente da patente. O processo de escolha seria por eliminação, sendo que o último soldado a ser selecionado se salvaria. O processo de eliminação consiste em: organizar os soldados em uma volta da fogueira; escolher um soldado para iniciar a contagem e sortear um único número. Ao final da contagem, o soldado escolhido seria eliminado e o processo seria reiniciado a partir do próximo soldado, até só restar o soldado ganhador.

Entradas:

- Nomes dos soldados que estão cercados;
- Opção de início de contagem:
 - i. Iniciar contagem a partir do primeiro soldado da lista;
 - ii. Iniciar contagem a partir de um soldado sorteado aleatoriamente da lista;
 - iii. Informar o nome do soldado para iniciar a contagem.

Saídas:

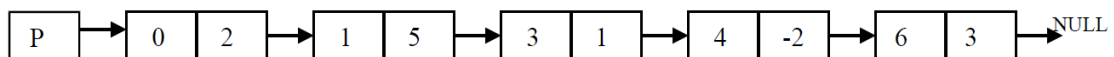
- No caso da opção de contagem (2), imprimir o nome do soldado sorteado;
- Imprimir o número sorteado;
- Imprimir os nomes dos soldados eliminados, na ordem de eliminação;
- Imprimir o nome do Sobrevivente.

Obs.: cabe ao discente escolher a **MELHOR** técnica de implementação para o problema.

8) Implementar um programa para manipulação de polinômios do tipo

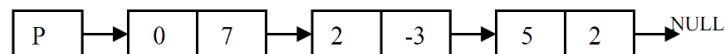
$$P(x) = a_n x^n + a_{(n-1)} x^{(n-1)} + \dots + a_1 x^1 + a_0$$

Para tal, o polinômio deve ser armazenado através de uma lista ordenada, sendo que cada elemento i da lista deve armazenar o k -ésimo termo do polinômio (diferente de 0), e deve conter o valor k da potência de x (inteiro) e o coeficiente a_k correspondente (inteiro). Por exemplo, o polinômio $P(x) = 3x^6 - 2x^4 + x^3 + 5x + 2$ deve ser representado pela lista:



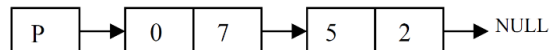
Fica a critério do discente a escolha da técnica de implementação, tanto em relação à forma de alocação, quanto à forma de agrupamento. A utilização de cabeçalho, encadeamento cíclico/simples/duplo também fica a critério do discente. Deve ser criada uma interface que permita ao usuário executar qualquer uma das operações a seguir, a qualquer momento:

- **Inicializar um polinômio:** Fazer $P(x) = 0x^0$
- **Inserir um novo termo $a_k x^k$ no polinômio existente:** Se já existe um termo $a_k x^k$ no polinômio o valor do coeficiente do novo termo a_k deve ser adicionado ao já existente a_k , assim: $P(x) = a_n x^n + \dots + (a_k + a_k') x^k + \dots + a_1 x^1 + a_0$
- **Imprimir $P(x)$:** Se o polinômio for $P(x) = 2x^5 - 3x^2 + 7$, a representação interna será:



A seguinte expressão deverá ser visualizada na tela: $+7 -3x^2 +2x^5$

- **Eliminar o termo associado à k -ésima potência:** Se o polinômio atual for $P(x) = 2x^5 - 3x^2 + 7$ (representação interna no exemplo acima) e o usuário solicitar a remoção do termo associado à potência 2 de x , o polinômio resultante será $P(x) = 2x^5 + 7$ e o nó referente à potência 2 de x deve ser liberado resultando na estrutura:



- **Reinicializar um polinômio:** Fazer $P(x) = 0x^0$ e liberar os nós do $P(x)$ anterior;
- **Calcular o valor de $P(x)$ para um valor de x solicitado:** Por exemplo, se o polinômio atual for $P(x) = 3x^6 - 2x^4 + x^3 + 5x + 2$ e o usuário solicitar o cálculo de $P(x)$ para $x = 2$, o valor de $P(2)$ deve ser calculado: $P(2) = 3(2)^6 - 2(2)^4 + (2)^3 + 5(2) + 2 = 3 \times 64 - 2 \times 16 + 1 \times 8 + 5 \times 2 + 2 = 180$ e o valor 180 deve ser apresentado na tela.