

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Curso de Bacharelado em Ciência da Computação
2º Trabalho de Algoritmos e Estrutura de Dados 1 - 2019/1
Profa. Gina M. B. Oliveira

- A entrega foi dividida em duas datas:
 - 1ª Entrega (24/6 até 23H59): entregar todos os exercícios da Parte A, exceto os marcados como Extra (ou seja, exercícios 1, 2, 3, 4, 8 e 9) e os exercícios 10 e 11 da PARTE B.
 - 2ª Entrega (1/7 até 13H00): aplicações mais elaboradas da PARTE B. Ou seja, exercícios 12, 13 e 14.
- A apresentação individual dos códigos será agendado posteriormente pela professora na semana de 1/7 a 3/7.
- Os códigos deverão ser implementados somente em Linguagem C, sendo necessária a utilização das estruturas de dados conforme discutidas em sala.

PARTE A – LISTAS (CÍCLICA E DUPLA), PILHAS, FILAS, FILAS DE PRIORIDADE e DEQUE: operações simples

- 1) (*) 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** e **imprime_lista**, além de incorporar as operações a seguir:
 - **Tamanho**: retorna o número de elementos da lista.
 - **Maior**: retorna o maior elemento da lista
- 2) (*) 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** e **imprime_lista**, além de incorporar as operações a seguir:
 - **Tamanho**: retorna o número de elementos da lista.
 - **Remover maior**: remove o maior elemento encontrado na lista.
- 3) (*) Implementar o TAD Pilha usando alocação **estática/seqüencial**. A TAD deve conter todas as operações vistas em sala e laboratório: **Inicializar_Pilha**, **Pilha_vazia**, **Pilha_cheia**, **Empilha**, **Desempilha**, **Le_topo**, **Imprimir (do topo para a base)**, além de incorporar as operações a seguir:
 - **Imprimir_reversa**: imprimir os elementos da pilha, da base para o topo.
 - **Palindrome**: verifica se uma string de entrada é uma palíndrome, usando uma pilha nessa verificação (obrigatório).
- 4) (*) Implementar o TAD Pilha usando alocação **dinâmica/encadeada**. A TAD deve conter todas as operações vistas em sala e laboratório: **Inicializar_Pilha**, **Pilha_vazia**, **Empilha**, **Desempilha**, **Lê_topo**, **Imprimir (do topo para a base)**, além de incorporar as operações a seguir:

- **Palindrome:** verifica se uma string de entrada é uma palíndrome, usando uma pilha nessa verificação (obrigatório).
 - **Pares_e_impares:** empilha uma sequência qualquer de inteiros positivos digitados pelo usuário. Quando o último elemento for inserido, o conteúdo da primeira pilha deve ser distribuído em outras duas pilhas: uma contendo os valores pares e outra contendo os valores ímpares.
- 5) (Extra) Implementar o TAD Fila usando alocação **estática/seqüencial circular (desprezo de uma posição do vetor)**. A TAD deve conter todas as operações vistas em sala e laboratório: Inicializar_Fila: Fila_é_vazia: Fila_é_cheia: Insere_fila; Remove_fila: Imprimir.
 - 6) (Extra) Implementar o TAD Fila usando alocação **estática/seqüencial circular (com contador de elementos)**. A TAD deve conter todas as operações vistas em sala e laboratório: Inicializar_Fila: Fila_é_vazia: Fila_é_cheia: Insere_fila; Remove_fila: Imprimir.
 - 7) (Extra) Implementar o TAD Fila usando alocação **dinâmica/encadeada simples (não circular)**. A TAD deve conter todas as operações vistas em sala e laboratório: Inicializar_Fila: Fila_é_vazia: Fila_é_cheia: Insere_fila; Remove_fila: Imprimir.
 - 8) (*) Implementar o TAD Fila de Prioridade Ascendente usando alocação **dinâmica/encadeada e inserção ordenada**. Operações que o TAD deve contemplar:
 - Inicializar_fpa, Fpa_é_vazia, Fpa_é_cheia, Insere_fpa, Remove_fpa
 - Imprimir: imprimir os elementos da fpa, do início para o final.
 - 9) (*) Implementar o TAD Deque usando alocação **dinâmica/duplamente encadeada**. Operações que o TAD deve contemplar:
 - Inicializar_Deque, Deque_é_vazia, Insere_início_deque, Insere_final_deque, Remove_início_deque, Remove_final_deque
 - Imprimir: imprimir os elementos da Deque, do início para o final.

PARTE B – APLICAÇÕES

- 10) (*) Implementar um programa que faça conversões de números inteiros na base 10 para outras bases, de acordo com a opção do usuário. Conversões que devem ser previstas: Decimal para Binário (implementada em sala) e Decimal para Octal. Utilizar a implementação do TAD Pilha usando alocação **estática/seqüencial**. Obs: Opcional: Fazer a conversão Decimal para Hexadecimal.
- 11) (*) Implementar um programa para manipulação de expressões matemáticas envolvendo variáveis literais de A a J, operadores (+ (adição), - (subtração), / (divisão), * (multiplicação), ^ (potenciação)) e os delimitadores de escopo tipo parênteses (“(”, ”)”). Utilizar a implementação do TAD Pilha usando alocação **dinâmica/encadeada**.

Para tal, o programa deve ter as seguintes funcionalidades:

- Entrada dos valores das literais: o usuário deve associar os valores a todas as literais de A a J.
- Entrada de expressões: o usuário deve optar entre entrar com expressões em 3 formatos: a) forma pós-fixa, b) forma infixa com uso de parênteses em todas as operações para indicar a precedência. c) forma infixa com uso de parênteses eventuais (quando necessário, para modificar a precedência da operação). Se o usuário optar pelo formato b) ou c), o programa deverá realizar a conversão da expressão para a forma pós-fixa e imprimir a expressão resultante da conversão.
- Avaliação da expressão: o programa deve avaliar a expressão digitada pelo usuário (após a conversão para a forma pós-fixa, se necessário), associando os valores das literais e imprimindo o resultado da expressão.

12)(**) Implementar o problema de Josephus utilizando o **TAD lista**. Cabe ao aluno escolher a técnica de implementação mais adequada para o problema.

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 um 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:
 - (1) Iniciar contagem a partir do primeiro soldado da lista.
 - (2) Iniciar contagem a partir de um soldado sorteado aleatoriamente da lista.
 - (3) Informar o nome do soldado para iniciar a contagem.

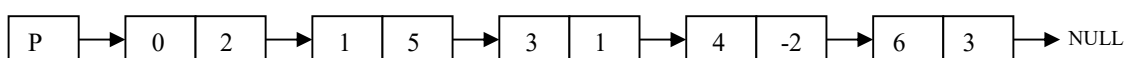
Saídas:

- 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.

13)(**) Implementar um programa para manipulação de polinômios do tipo

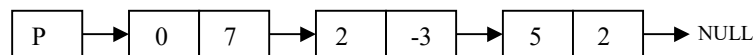
$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x^1 + a_0$$

Para tal, o polinômio deve ser armazenado através de uma TAD **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:



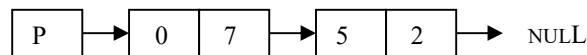
Fica a critério do aluno a escolha da técnica de implementação (a utilização de cabeçalho, encadeamento cíclico/simples/duplo também fica a critério do aluno). Deve ser criada uma interface que permita ao usuário executar qualquer uma das operações a seguir, a qualquer momento:

- **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.

14) (**) Enunciado do Dilema do Fazendeiro: “Um fazendeiro encontra-se na margem norte de um rio, levando consigo um lobo, uma cabra e um repolho. O fazendeiro precisa atingir a outra margem do rio (sul) com toda a sua carga intacta. Para isso dispõe somente de um pequeno bote com capacidade para levar apenas ele mesmo (sozinho) ou ele com APENAS mais uma de suas cargas. O fazendeiro poderia cruzar o rio quantas vezes fossem necessárias para transportar seus pertences, mas o problema é que, na ausência do fazendeiro, o lobo pode comer a cabra e essa, por sua vez, pode comer o repolho. O objetivo é encontrar uma sequência de passos (travessias) que resolva esse dilema.”

- a) Partindo da configuração inicial (0,0,0,0) (que representa todos elementos na margem Norte do rio), utilize uma estrutura do tipo FILA para armazenar todas as soluções parciais (devem representar a configuração intermediária e uma lista com todos os passos executados até atingir essa configuração) até que se chegue em uma solução final (1,1,1,1) (atinge a configuração final que representa todos elementos na margem Sul do rio e a lista com todos os passos executados para atingir essa configuração). Para isso, considere, que as 4 travessias possíveis a cada passo são: (1) Fazendeiro atravessa sozinho, (2) Fazendeiro atravessa com a cabra, (3) Fazendeiro atravessa com o lobo, (4)

(2) Fazendeiro atravessa com o repolho. A cada exploração de uma solução parcial que é retirada da fila, deve-se checar quais travessias são possíveis (eliminar a travessia que refaz o último passo para eliminar ciclos). A cada passo da exploração imprima: O número do passo de exploração (1, 2, 3, ...), Estado inicial da Fila, Nó a ser explorado, Fila após a remoção do nó, Novas soluções parciais encontradas no passo. Utilize a estrutura de dados discutida em sala de aula para representar as soluções parciais. Utilizar a implementação do TAD Fila usando alocação **dinâmica/encadeada simples (não circular)**. Utilizar a implementação do TAD Lista usando alocação **estática/sequencial**.

- b) Repita o item a, verificando o que acontece quando uma PILHA é utilizada ao invés da FILA (na armazenagem das soluções não exploradas). Utilizar a implementação do TAD Pilha usando alocação **dinâmica/encadeada**. Utilizar a implementação do TAD Lista usando alocação **estática/sequencial**.