ESCOLA POLITÉCNICA DE PERNAMBUCO – POLI
ENGENHARIA DA COMPUTAÇÃO - ECOMP
ESTRUTURAS DE DADOS - 2019.2

Lista de exercícios 04 - Pilhas
13 de setembro de 2019

OBS: Para cada questão solucionada, envie em seu projeto os testes efetuados em um método main.

(ESCOLHA 10 QUESTÕES PARA RESPONDER)

1) Implemente um algoritmo que armazena cadeias de texto de forma sequencial. Ele deverá ter a possibilidade de adicionar novas alterações (sejam letras, números, símbolos) bem como a opção de "desfazer" uma dada operação. **Use somente a estrutura Pilha** para auxiliar na manipulação dos dados. Construa os seguintes métodos (confira o exemplo abaixo):

OPERAÇÃO	CONTEÚDO DA PILHA
Adicionar "Estudando"	**Topo** -> "Estudando"
Adicionar "_Pilhas"	**Topo** -> "Estudando_Pilhas" -> "Estudando"
Adicionar "_agora!"	**Topo** -> "Estudando_Pilhas_agora!" "Estudando_Pilhas" -> "Estudando"
Desfazer operação	**Topo** ->"Estudando_Pilhas" -> "Estudando"

a) Adicionar nova porção de texto ao texto existente;
b) Desfazer a edição mais recente, voltando ao estado anterior;
c) Listar a fila de alterações salvas (você não deve utilizar outro TAD senão o tipo Pilha);
d) Informe quantas alterações já foram salvas na estrutura;

2) Modele um TAD Pilha de livros(nome, anoDeLancamento) com seus métodos usuais. Imagine que seja necessário retirar um livro dessa pilha, que não está, necessariamente, no topo da pilha. Desenvolva um método para isso. Você não pode remover livros com outros em cima! Use as operações de manipulação e a estrutura pilha para resolver.

3) Desenvolva um algoritmo para testar se uma pilha P1 existe dentro de uma pilha P2. **Obs.: você deve utilizar apenas os métodos push, pop, peek e size da classe Pilha para fazer tal procedimento.**
-Considere que P1 e P2 já existem.
-Considere a existência de pilhas que não sejam equivalentes desde o ínicio da pilha P2, de pilhas que não chegam até o final de P2, de pilhas que estejam presentes no meio da pilha P2 e de pilhas exatamente iguais, desde que tenham a mesma ordem.

4) Implemente uma pilha estática de pilha, ou seja, uma "pilha que empilha pilhas". Cada pilha dentro da pilha principal é de inteiros e também estática, e todas têm 10 espaços de memória. Implemente:

(a) - *pop()*, que funciona tradicionalmente, removendo e retornando o último elemento inserido.
Obs.: Em caso da pilha principal sofrer *underflow*, retorne 0.

(b) - *sum()*, que retornará uma pilha de inteiros que em cada valor será a soma dos valores de uma das pilhas da pilha principal.
Obs.: O método não deve modificar a pilha.

(c) - *push(*int e*)*, que tentará inserir na primeira pilha, e, em caso de estar cheia, irá inserir na segunda, e assim sucessivamente.
Obs.: Em caso da pilha principal sofrer *overflow*, some todos os valores da pilha como o valor inicial da primeira pilha, e insira o valor "e" normalmente.

(d) - *toString()*, que retornará uma String que mostre a pilha principal no seguinte modelo:

Ex.: ->(val1, val2, val3,...)->(val11, val12,...)->..., onde os parênteses denotam as pilhas "interiores". Também não deve modificar a pilha.

5) Implemente um TAD Lista simplesmente encadeada com os métodos add(T e), remove(T e) e toString() **recursivos**. Use métodos auxiliares se necessário. Não é preciso implementar os outros métodos usuais de Lista.

6) Dado um TAD Pilha, implemente um algoritmo que remova os **elementos de chave ímpar** e adicione-os em um TAD Fila. Não é válido manipular os dados da estrutura Pilha utilizando nós auxiliares, mas sim Pilhas auxiliares. A pilha de entrada, com os elementos removidos, não deve ter sua referência inicial alterada.

a) Efetue a remoção dos elementos de chaves ímpares;
b) Imprima os valores contidos em ambas as estruturas.

7) Implemente uma pilha usando uma fila para armazenar seus dados. A fila deve ser acessada apenas por seus métodos, não pelos ponteiros, e ambas devem ter todos os métodos usuais.

8) Desenvolva o método div(int m, int n), recursivo, que retorna a divisão inteira de m por n. **Não use as operações aritméticas de multiplicação, divisão e resto (*, / e %) neste exercício**.

9) Construa um método que recebe uma lista encadeada de números inteiros e retorna uma lista sem repetições, ou seja, uma lista onde cada número apareça apenas uma vez. **Utilize apenas os métodos usuais da lista.**
Exemplo:
12 5 -7 8 5 9 12 1 8 → 12 5 -7 8 9 1

10) Dada uma lista encadeada de caracteres formada por uma seqüência alternada de letras e dígitos, construa um método que retorne uma lista na qual as letras são mantidas na seqüência original e os dígitos são colocados na ordem inversa.
Exemplos:
A 1 E 5 T 7 W 8 G → A E T W G 8 7 5 1
3 C 9 H 4 Q 6 → C H Q 6 4 9 3

Como mostram os exemplos, **as letras devem ser mostradas primeiro, seguidas dos dígitos.**
Dica: Você pode utilizar mais de uma estrutura de dados.

11) Escreva um programa que simule o controle de uma pista de decolagem de aviões em um aeroporto. Neste programa, o usuário deve ser capaz de realizar as seguintes tarefas:
a) Listar os aviões aguardando na fila de decolagem;
b) Autorizar a decolagem do primeiro avião da fila;
c) Adicionar um avião à fila de espera;
d) Listar todos os aviões na fila de espera;
e) Mover um avião da fila de espera para a fila de decolagem.