EXERCÍCIOS DE PREPARAÇÃO PARA PROVA 1 – ESTRUTURA DE DADOS 1
PROF. IGOR MACHADO COELHO 2018/1 – Ccomp
DEVE SER ENTREGUE NA P1 (VALE 1PT NA PROVA!)

TODOS CÓDIGOS DEVEM SER IMPLEMENTADOS EM C/C++, COM A RESPECTIVA ANÁLISE DE COMPLEXIDADE (PIOR E MELHOR CASOS)

1. Um deque geral deve prover as seguintes operações para: inserir no início, inserir no fim, remover no início, remover no fim, obter primeiro elemento, obter último elemento.

Defina o estrutura Deque e escreva os procedimentos e funções acima de forma análoga ao realizado com Pilhas e Filas. Considere tanto o uso da alocação sequencial quanto da encadeada.

2. Reescreva as operações de Pilha utilizando duas Filas como estrutura de dados auxiliar para guardar os elementos.

3. Reescreva as operações de Fila utilizando duas Pilhas como estrutura de dados auxiliar para guardar os elementos.

4. Escreva um algoritmo que dada uma pilha P, inverta a ordem dos elementos de P. Seu algoritmo deve usar apenas espaço auxiliar constante e:
a. uma fila
b. duas pilhas
c. uma pilha

5. Escreva um algoritmo que dada uma fila F, inverta a ordem dos elementos de F. Seu algoritmo deve usar apenas espaço auxiliar constante e:
a. uma pilha
b. duas filas

6. Seja X um arquivo em disco que guarda uma sequência de N naturais. Sabendo-se que o conteúdo de X é muito grande para ser carregado todo em memória, faça um algoritmo que escreva os últimos 1000 naturais de X em tempo $\theta(N)$. Assuma que o arquivo deve ser acessado sequencialmente, da seguinte maneira: ele deve ser primeiramente aberto, depois cada os naturais são lidos um a um até que um -1 seja lido (indicando final de arquivo), situação em que o arquivo deve ser fechado. Como acesso a disco é uma operação custosa, deseja-se fazer tal impressão numa única passagem pelos dados do arquivo.

7. Criar uma variação de pilha, chamada de PilhaMin, que, além de fornecer as operações de pilha em tempo constante, define uma operação que retorna o elemento de P com a menor chave em tempo constante

8. Escrever um algoritmo que converta uma expressão aritmética parentizada usando as 4 operações para a expressão correspondente em notação polonesa inversa, na qual quando um operador é encontrado, é aplicado imediatamente aos dois operandos que o antecedem. Ex: ((A+B)*(C-D)) é expresso como AB+CD-* em notação polonesa inversa.

9. Utilizando uma pilha, escreva um algoritmo que compute uma expressão escrita na em notação polonesa inversa (note que na notação polonesa inversa não são necessários parênteses). Ex: 5 + (1 + 2) × 4 – 3 é representada por 5 1 2 + 4 × + 3 -

10. Uma pilha é ordenada se os elementos são removidos em ordem crescente. Dada uma pilha P, escreva um algoritmo que remova e reinsira elementos de P (através das

funções empilha e desempilha) até que P se torne ordenada. Seu algoritmo deverá utilizar espaço auxiliar constante acrescido de outra pilha.

11. Escrever um algoritmo que verifique se uma expressão está corretamente parentizada (isto é, sem que um parênteses abra (respectivamente feche) sem fechar (respectivamente sem abrir) entre um par de abre-e-fecha de parênteses). Você deve atender as seguintes especificações:
a. não há limite de tamanho para a entrada (exceto pela memória disponível)
b. existem vários tipos de parênteses. Cada fechamento de parênteses deve corresponder a uma abertura do mesmo tipo. O tipo é especificado pelo nome logo após cada parêntese.
Exemplo de entrada:
(p(p)p(p)p(p(p)p)p)p ⇒ OK (só um tipo de parênteses, p)
(p1(p2)p1)p2 ⇒ incorreto (dois tipos de parênteses, p1 e p2)
(p1(p2)p2)p1 ⇒ OK (dois tipos de parênteses, p1 e p2)

12. Projete uma estrutura chamada PilhaDividida, que funciona como uma pilha (topo, push, pop), mas internamente é composta por T pilhas de tamanho limitado K. Além das pilhas internas, você pode usar espaço auxiliar constante. Projete as operações de push, pop e topo na PilhaDividida.

13. Projete um programa para monitorar entradas e saídas de pacotes em um roteador. O programa deve ler uma sequência de N pares X Y onde X = "E" representando se o roteador deve receber em seu buffer um novo dado cujo valor é Y, ou X="S" representando que ele deve enviar o dado mais antigo no buffer pelo canal Y. Como entrada, haverá o valor de N seguido dos N pares X Y. Como saída, o programa deve imprimir, tão logo um par X Y com X = "S" é lido, o par R S representando que o dado R será enviado pelo canal S. O algoritmo deve ter complexidade de tempo O(N). Exemplo:

Entrada:
13
E 10 E 2 S 4 E 3 E 7 S 5 S 6 S 7 E 6 E 1 E 4 E 5 S 1

Saída:
10 4
2 5
3 6
7 7
6 1