Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Departamento Acadêmico de Computação

ESTRUTURA DE DADOS - C31 – Engenharia de Computação
Prof. Danilo Sipoli Sanches

Lista de Exercícios 1

1. Faça uma função que receba uma lista encadeada L1 e retorne uma **nova lista do tipo duplamente encadeada LD1** com os elementos pares existentes em L1;

2. Uma estrutura DEQUE é um TAD Fila em que elementos podem ser inseridos e retirados de ambas as extremidades da estrutura (estática ou dinâmica) APENAS. Outros tipos de inserção e remoção (em outras posições) não são válidos. Escreva um programa com as funções necessárias para implementar um DEQUE representado por uma lista duplamente encadeada.

3. Faça uma rotina para verificar se os elementos de uma **fila** (**representada por listas encadeadas**) estão ordenados de forma crescente.

4. Faça um programa para testar se duas pilhas (representadas por lista encadeadas) P1 e P2 são iguais;

5. Faça uma lista duplamente encadeada circular e implemente as seguintes funções para ela:
   a-)Insere
   b-)Remove
   c-)Imprime

6. Construa uma função para remover todos os elementos pares de uma **lista duplamente encadeada circular**. Obs: implementar as funções necessárias para inserção, remoção e impressão da lista duplamente encadeada circular.

7. Construa uma função que retorne a quantidade de números primos em uma lista duplamente encadeada. Obs: implementar as funções necessárias para inserção, remoção e impressão da lista duplamente encadeada.

8. Faça uma função que receba uma lista encadeada L1 e retorne a lista L1 com os elementos invertidos. Para isso, deverá ser utilizada uma pilha. Exemplo: Lista1 = {1,2,3}, após passar pela função a Lista1 ficará: Lista1 = {3,2,1}.

9. Faça um programa que verifique se duas filas dinâmicas são iguais. Obs:implementar as funções necessárias para inserção, remoção e impressão da fila dinâmica.

10. Faça uma função que receba uma pilha P1 e retorne essa mesma pilha com os elementos invertidos.

11. Considere uma lista duplamente encadeada para armazenar números inteiros. Implemente uma função que receba como parâmetros uma lista com seus elementos ordenados em ordem crescente e um número inteiro *x*, e insira um novo nó na lista com o valor *x*, preservando a ordenação da lista.

12. Faça uma **Fila** representada por uma lista encadeada circular e implemente as seguintes funções para ela:
    a-)Insere
    b-)Remove
    c-)Imprime

13. Faça uma **Pilha** representada por uma lista encadeada circular e implemente as seguintes funções para ela:
    a-)Insere
    b-)Remove
    c-)Imprime

14. Crie um pilha representada por lista encadeada> Após isso, faça uma função que remova as elementos ímpares desta pilha e armazene em uma lista encadeada circular.

15. Crie um lista duplamente encadeada. Após isso, faça uma função que remova as elementos ímpares desta lista e armazene em uma Fila representada por uma lista encadeada circular.

16. Faça uma função que receba uma Fila F1 e retorne essa mesma Fila com os elementos invertidos.

17. Escreva uma função em C para trocar os elementos *m* e *n* de uma lista simplesmente encadeada (*m* e *n* podem ser chaves ou mesmo ponteiros para os elementos – a escolha é sua).

18. Agora escreva uma função em C para trocar os elementos *m* e *n* de uma lista duplamente encadeada (*m* e *n* podem ser chaves ou mesmo ponteiros para os elementos – a escolha é sua).

19. Escreva uma rotina, inssub(l1,i1,l2,i2,len) para inserir os elementos da lista *l2*, começando no elemento *i2* e continuando por *len* elementos na lista *l1*, começando na posição i1. Nenhum elemento da lista l1 deverá ser removido ou substituído. Se i1 > length(l1) + 1 (onde length(l1) indica o número de nós na lista), ou se i2 + len – 1 > length(l2), ou se i1 < 1, ou se i2 < 1, imprima uma mensagem de erro. A lista l2 deve permanecer inalterada.

20. Escreva uma função em C, search(l,x), que receba um ponteiro l para uma lista de inteiros e um inteiro x, e retorne um ponteiro para um nó contendo x, se existir, e o ponteiro nulo, caso contrário. Escreva outra função, searchinsert(l,x), que inclua x em l se ele não for encontrado, e retorne sempre um ponteiro para um nó contendo x.