SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS V - DIVINÓPOLIS

Lista de exercícios (3 pontos extras)

Aluno: _____

Professor: Eduardo Habib Bechelane Maia **Nota:**_____

Entregar escrito à mão logo antes da prova 4

1. Considere que um banco possui uma lista de clientes e cada cliente possui uma conta corrente onde deve ser possível armazenar o número da conta, CPF e nome do correntista, limite da conta (para crédito especial) e o saldo:

   Tendo em vista que:
   a. As operações possíveis na conta são: depósito, retirada e impressão de extrato.
      i. O extrato exibe o número da conta, nome do cliente, transações, o saldo, limite da conta;

Faça um programa que crie uma estrutura para representar o tipo conta. A estrutura deve conter todos os dados da conta. Em seguida, supondo que um banco possa possuir até 1000 clientes, implemente as operações básicas da conta (em funções) e faça um main que possua um switch case que teste o cadastro de clientes, depósito, retirada e impressão do extrato de uma conta.

2. Dado o trecho de código abaixo:

   ```
   int x, *px;
   px = &x;
   ```

   Explique sucintamente a diferença entre utilizar px, *px, &x e x. Utilize exemplos, se necessário, para ilustrar a explicação de forma mais clara.

3. Explique as estruturas de dados Lista, Fila e Pilha, e forneça pelo menos 2 exemplos reais de utilização para cada uma delas.

4. A sequência de Fibonacci começa com os números 1 e 1 e, em seguida, cada novo número da sequência é a soma dos dois números imediatamente anteriores, como se vê a seguir:

   $$1, 1, \underbrace{2}_{1+1}, \underbrace{3}_{1+2}, \underbrace{5}_{2+3}, \underbrace{8}_{3+5}, \underbrace{13}_{5+8}, \underbrace{21}_{8+13}, \ldots$$

   Crie uma função **RECURSIVA** que receba um número ‘n’ inteiro maior que zero e retorne o enésimo número de Fibonacci.

5. Escreva um programa em C que crie e manipule uma agenda de contatos. Cada contato deve ter as seguintes informações: nome, e-mail e telefone. O programa deve usar uma struct para representar um contato e uma lista encadeada para armazenar os contatos. O programa deve ter um menu com as seguintes opções:
   a. Criar a struct contato com os campos nome, e-mail e telefone. Definir uma lista de contatos em uma estrutura.
   b. Inserir um novo contato na agenda. O programa deve pedir ao usuário os dados do contato e inseri-los no início da lista. O programa deve usar uma função para fazer a inserção, que recebe como parâmetro a lista de contatos e os dados a serem inseridos.
   c. Listar todos os contatos da agenda. O programa deve mostrar na tela os dados de todos os contatos da lista, um por linha. O programa deve usar uma função para fazer a listagem, que recebe como parâmetro a lista de contatos.

d. Pesquisar um contato na agenda pelo nome. O programa deve pedir ao usuário o nome do contato que deseja pesquisar e mostrar na tela os dados do contato encontrado, ou uma mensagem informando que o contato não foi encontrado. O programa deve usar uma função para fazer a pesquisa, que recebe como parâmetros a lista de contatos onde a pesquisa será feita e o nome do contato a ser pesquisado.
e. Sair do programa. O programa deve encerrar sua execução se o usuário digitar essa opção.
f. Explique a diferença entre passagem de valor por parâmetro e passagem de referência por parâmetro em uma função. Dê exemplos de código em C que ilustrem as vantagens e desvantagens de cada método de passagem de parâmetro.
g. Diferencie as operações de inserção e remoção em uma Lista, uma Fila e uma Pilha, e indique em que situações cada uma dessas estruturas de dados é mais adequada para representar uma coleção de elementos. Dê exemplos de aplicações que usam essas estruturas de dados.
h. Escreva uma função recursiva que receba um número inteiro positivo n e retorne o número de dígitos que ele possui. Por exemplo, se n = 123, a função deve retornar 3, que é o número de dígitos de n.

6. Escreva um programa em C que crie e manipule uma coleção de cartas. Cada carta deve ter as seguintes informações: naipe, valor e cor. O programa deve usar uma struct para representar uma carta e uma pilha para armazenar as cartas. O programa deve ter um menu com as seguintes opções:
   a. Criar a struct carta com os campos naipe, valor e cor. Definir uma pilha de cartas em uma estrutura.
   b. Empilhar uma nova carta na coleção. O programa deve pedir ao usuário os dados da carta e empilhá-la no topo da pilha. O programa deve usar uma função para empilhar, que recebe como parâmetro a pilha de cartas e a carta a ser empilhada.

   c. Desempilhar a carta do topo da coleção. O programa deve mostrar na tela os dados da carta que está no topo da pilha e removê-la da pilha. O programa deve usar uma função para desempilhar, que recebe como parâmetro a pilha de cartas e retorna os dados da carta removida.

   d. Verificar se a pilha está vazia ou cheia. O programa deve informar ao usuário se a pilha de cartas está vazia ou cheia, ou seja, se não há nenhuma carta na pilha ou se não há mais espaço para empilhar novas cartas. O programa deve usar uma função para fazer a verificação, que recebe como parâmetro a pilha de cartas e indica se a pilha está vazia ou cheia.

   e. Sair do programa. O programa deve encerrar sua execução se o usuário digitar essa opção.
   OBS 1: A pilha de cartas deve ter um tamanho máximo definido previamente.
   OBS 2: Você pode usar parâmetros adicionais nas funções se julgar necessário.

7. Dado o trecho de código abaixo:

   char *str = “Hello”; str[0] = ‘h’;

   Explique por que esse código pode causar um erro de segmentação e como corrigi-lo. Utilize exemplos, se necessário, para ilustrar a explicação de forma mais clara.

a. Escreva uma função recursiva em C que receba um número inteiro positivo n e retorne a soma dos dígitos desse número. Por exemplo, se n = 123, a função deve retornar 6. Não utilize laços nem variáveis globais na sua solução.

Escreva um programa em C que crie e manipule uma lista de tarefas. Cada tarefa deve ter as seguintes informações: título, descrição, e data de vencimento (dia, mês, ano). O programa deve usar uma struct para

representar uma tarefa e uma lista duplamente encadeada para armazenar as tarefas na lista. O programa deve ter um menu com as seguintes opções:

a. Criar a struct **Tarefa** com os campos título, descrição, e data de vencimento. Definir uma lista duplamente encadeada de tarefas em uma estrutura.

b. Inserir uma nova tarefa no início da lista de tarefas. O programa deve pedir ao usuário os dados da tarefa (título, descrição, data de vencimento) e inseri-los no início da lista. O programa deve usar uma função para fazer a inserção, que recebe como parâmetro a lista de tarefas e os dados a serem inseridos.

c. Inserir uma nova tarefa após uma tarefa específica na lista. O programa deve pedir ao usuário o título da tarefa que será a referência e os dados da nova tarefa (título, descrição, data de vencimento). A nova tarefa deve ser inserida logo após a tarefa de referência. Se a tarefa de referência não for encontrada, o programa deve informar ao usuário. O programa deve usar uma função para fazer a inserção, que recebe como parâmetros a lista de tarefas, o título da tarefa de referência, e os dados da nova tarefa.

d. Remover uma tarefa específica da lista. O programa deve pedir ao usuário o título da tarefa a ser removida. Se a tarefa for encontrada, o programa deve removê-la da lista e liberar a memória alocada para ela. Se a tarefa não for encontrada, o programa deve informar ao usuário. O programa deve usar uma função para realizar a remoção, que recebe como parâmetros a lista de tarefas e o título da tarefa a ser removida.

e. Listar todas as tarefas na ordem atual. O programa deve mostrar na tela os dados de todas as tarefas da lista, uma por linha (título, descrição e data de vencimento). O programa deve usar uma função para fazer a listagem, que recebe como parâmetro a lista de tarefas.

f. Pesquisar uma tarefa na lista pelo título. O programa deve pedir ao usuário o título da tarefa que deseja pesquisar e mostrar na tela os dados da tarefa encontrada (título, descrição, data de vencimento), ou uma mensagem informando que a tarefa não foi encontrada. O programa deve usar uma função para fazer a pesquisa, que recebe como parâmetros a lista de tarefas onde a pesquisa será feita e o título da tarefa a ser pesquisada.

g. Verificar se a lista de tarefas está vazia. O programa deve informar ao usuário se a lista de tarefas está vazia, ou seja, se não há nenhuma tarefa na lista. O programa deve usar uma função para fazer a verificação, que recebe como parâmetro a lista de tarefas.

h. Sair do programa. O programa deve encerrar sua execução se o usuário digitar essa opção.

Observações:

- **Data de Vencimento:** O campo de data de vencimento pode ser armazenado como três inteiros separados (dia, mês, ano) ou como uma string formatada.
- **Desafios Adicionais:** Valendo um ponto a mais além dos 3 pontos extras, implemente uma funcionalidade para ordenar as tarefas por data de vencimento, e outra para editar uma tarefa existente.