Instruções gerais:

- O prazo de entrega do teste será até o dia 17/07/2024 (quarta-feira) às 23h59 e deverá ser enviado através deste link.
- Para envio das respostas das perguntas teóricas, solicitamos que façam uma cópia deste documento, incluindo seu nome e sobrenome no título.
- As perguntas que requeiram elaboração de código devem acompanhar o código fonte em anexo, seja o arquivo diretamente em anexo ou link para o repositório.
    - Caso sua solução necessite de condições especiais para compilação e/ou execução, favor incluir as instruções.

Questão 1:

A empresa SmartWork trabalha com múltiplos tipos de profissionais diferentes. No momento, eles podem aderir a um e apenas um dos seguintes tipos de contrato:

1. Funcionários CLT, que recebem um salário fixo mensal
2. Profissionais terceirizados, os quais recebem um valor fixo pago por hora trabalhada no mês.

Com o intuito de motivar seus colaboradores, a empresa decidiu criar uma política de remuneração por comissão, pagando um bônus fixo por contrato assinado, podendo esta ser adotada tanto por funcionários CLT quanto terceirizados.

Exemplo:

João
Contrato: Terceirizado
Valor por hora: R$ 50,00
Horas trabalhadas 100
Total à receber: R$ 5000,00

Marcela:
Contrato: CLT
Salário: R$ 3000,00
Contratos assinados: 10
Comissão por contrato: R$ 200,00

Total à receber: R$ 5000,00

Joaquim:
Contrato: Terceirizado
Valor por hora: R$ 30,00
Horas trabalhadas 180
Contratos assinados: 12
Comissão por contrato: R$ 200,00
Total à receber: R$ 7800,00

Custo total da folha de pagamento da empresa: R$ 17800,00

Objetivo:

Sua missão é desenvolver um conjunto de rotinas em **C** que permitam:

1. Cadastrar as informações do mês de um número arbitrário de funcionários
2. Calcular individualmente o total à pagar para cada funcionário
3. Calcular o total da folha de pagamento da empresa.

Observação:

Não é necessário desenvolver uma interface com o usuário. As informações dos funcionários e as chamadas para as rotinas de cálculo podem ser chamadas diretamente na main. O objetivo deste exercício é criar as rotinas necessárias para gerar as informações, e não a forma como serão consumidas.

Informações adicionais:

Deseja-se uma implementação enxuta das informações dos funcionários. Por exemplo, se um funcionário não é pago por hora, não há necessidade de armazenar as informações de hora trabalhada e taxa por hora. Quanto menos informações redundantes ou não utilizadas, melhor.

De atenção à manutenção do código, evite código e dados duplicados e redundantes na implementação.

Questão 2:

Explique, em linhas gerais, como você faria para implementar uma rotina de testes automatizados para o problema anterior.

Eu dividiria em 3 testes separados:

1. Teste da função “cadastrarFuncionario” - Validar se ela está adicionando todos os dados do funcionário de forma correta.

2. Teste da função “calcularPagamentoAUmFuncionario” - Conferir se o valor calculado está de acordo com a equação proposta no problema.

3. Testa da função “calcularFolhaDePagamento” - Conferir se o valor calculado é de fato a soma do valor à pagar para cada funcionário.

Questão 3:

A empresa SmartWork, do problema 1, agora deseja implementar a possibilidade de um bônus anual para o funcionário, e a possibilidade de hora extra para os funcionários CLT. Explique em linhas gerais como faria a implementação desses dois itens.

Eu adicionaria ao struct “Funcionario” os campos “horas_extras” e “valor_hora_extra”. Uma nova função para adicionar horas extras a um determinado funcionário pelo id dele, conferindo se a variável “eTercerizado” é falsa.

E alteraria a função “calcularPagamentoAUmFuncionario” para receber o parâmetro “bonus_anual”, e realizar o cálculo levando em consideração os campos “horas_extras”, “valor_hora_extra” e o parâmetro “bonus_anual”.

Questão 4:

Explique o que são ponteiros para função (function pointer). Em quais ocasiões eles são úteis? Se possível, cite exemplos.

Os ponteiros para função são uma forma de apontar para um endereço de memória de uma função. Eles são úteis para armazenar, passar e chamar funções como se fossem variáveis.
Esse recurso traz bastante flexibilidade, podendo passar funções como parâmetros de outras funções, assim como eficiência, já que esses ponteiros eliminam a necessidade de instruções condicionais complexas para determinadas ações. A abstração também é maior, já que podem ser criadas

funções genéricas para operar funções diferentes sem compreender suas implementações.
A soma de todas essas coisas torna o código mais adaptável e dinâmico, podendo criar até mesmo algumas estruturas, como árvores de decisão, utilizando esses ponteiros para função.

Um exemplo simples é a criação de operações de ordenação personalizadas, onde você pode ter n funções de ordenação e uma única função "ordenar" que recebe como parâmetro o ponteiro para a função de ordenação que você deseja utilizar.

Questão 5:

Quais são os "tipos" de memória (RAM, etc.) tipicamente encontrados em microcontroladores? Quais seus papéis e relevância no funcionamento do MCU?

Memória RAM (Random Access Memory): Essa é a memória onde ficam os dados e instruções que estão sendo utilizados no momento, e por isso, ela costuma operar numa velocidade mais a próxima do processador, porém, tem menos espaço que a memória principal e é volátil, o que significa que caso ela fique sem energia, os dados são perdidos.

Memória Flash: Essa é a memória principal, é onde é armazenado o firmware e os dados de configuração. Ela não é volátil, então mesmo sem energia, os dados permanecem nela.

Memória EEPROM (Electrically Erasable Programmable Read-Only Memory): Essa memória também é não volátil, mas diferentemente da memória flash, ela permite a leitura e gravação de dados individualmente, e normalmente é utilizada para gravar dados de configuração personalizados para cada usuário. Vale ressaltar que deve-se guardar o mínimo de dados nela, pois ela tem um limite de gravações que podem ser feitas.

Memória ROM (Read-Only Memory): Diferentemente das outras memórias, essa só permite a leitura dos dados que estão nela. Ela é utilizada para armazenar os dados e o código de inicialização (bootloader) do MCU.

Memória Cache: Essa memória é volátil, assim como a RAM, e armazena cópias de dados frequentemente utilizados pelo processador, reduzindo o tempo de acesso à memória. Ela normalmente é ainda menor que a RAM, porém mais rápida que a RAM.