Universidade Federal de Mato Grosso

Bacharelado em Ciência da Computação

Semestre: 2024/1

Programação 2

Lista de Revisão

Exercício 1

Você foi designado para implementar um balanceador de carga simples em C++ que suporta três estratégias de distribuição: round robin, weighted round robin e random. O programa deve aceitar como entrada o número de servidores, seus pesos (para a estratégia weighted round robin) e o número de requisições que serão injetadas. Como saída, deve exibir quantas requisições cada servidor atendeu para cada uma das estratégias implementadas.

Estratégias

1. **Round Robin:** Esta estratégia distribui as requisições de forma circular entre os servidores. Cada requisição subsequente é atribuída ao próximo servidor na lista.
2. **Weighted Round Robin:** Nesta estratégia, cada servidor tem um peso associado que determina quantas vezes ele é escolhido antes de passar para o próximo na lista. Por exemplo, se um servidor tem peso 2 e outro tem peso 1, o primeiro servidor receberá duas vezes mais requisições do que o segundo em cada ciclo completo.
3. **Random:** Esta estratégia seleciona aleatoriamente um servidor para atender cada requisição.

Instruções para o Exercício

1. **Defina a Estrutura do Servidor:** Crie uma estrutura `Servidor` com campos para o nome, peso e número de solicitações atendidas.
2. **Implemente Round Robin:**
   - Crie uma função `roundRobin` que distribua as solicitações entre os servidores de maneira circular.
   - Registre o número de requisições atendidas por cada servidor.
3. **Implemente Weighted Round Robin:**
   - Crie uma função `weightedRoundRobin` que distribua as solicitações levando em conta os pesos dos servidores.
   - Os servidores com maior peso devem atender mais solicitações.
   - Registre o número de requisições atendidas por cada servidor.
4. **Implemente Random:**
   - Crie uma função `randomStrategy` que distribua as solicitações para servidores escolhidos aleatoriamente.
   - Registre o número de requisições atendidas por cada servidor.

Exercício 2

O problema de verificação de anagramas envolve determinar se duas strings dadas são anagramas uma da outra. Duas strings são consideradas anagramas se uma puder ser rearranjada para formar a outra.

Instruções

- **Entrada:**
  Duas strings para verificar se são anagramas uma da outra.
- **Saída:**
  Indicar se as duas strings são anagramas.
  Se forem anagramas, exibir uma mensagem indicando que são anagramas.
  Caso contrário, indicar que não são anagramas.
- **Exemplo:**
  As strings abaixo são anagramas uma da outra:
  - String 1: "listen"
  - String 2: "silent"

Exercício 3

O problema de validação de palíndromos envolve verificar se uma string é idêntica à sua versão invertida. Ou seja, uma string é um palíndromo se, ao ser lida de trás para frente, ela continua a mesma.

Instruções

- **Entrada:**
  Uma string para verificar se é um palíndromo.
- **Saída:**
  Indicar se a string é um palíndromo ou não.
- **Exemplo:**
  Suponha que temos as seguintes strings para validar como palíndromos:
  - String 1: "radar"
  - String 2: "abcba"
  - String 3: "hello"
  - String 4: "madam"

  - A saída esperada seria algo como:

```
"radar" é um palíndromo.
"abcba" é um palíndromo.
"hello" não é um palíndromo.
"madam" é um palíndromo.
```