**1. Efetue a leitura do material de apoio (Manual de Referência do**

**FreeRTOS v9.0.0 disponibilizado no Blackboard) e responda:**

**a. Responda com suas palavras para que serve a estrutura**

**semáforo?**

R:O semáforo é uma estrutura intrínseca para a computação, esta permite controlar o acesso a recursos compartilhados, evitar conflitos entre processos ou threads e controlar o fluxo de execução de um programa.

**b. Qual a diferença entre os semáforos binários e o mutexes?**

R: Referente ao indagado, no que diz respeito aos mutexes, estes são mais adequados para controlar o acesso a recursos compartilhados que podem ser usados por várias tarefas, mas apenas um número limitado de tarefas a cada vez. Em contrate, os semáforos binários só possuem dois valores para o gerenciamento, estes são, “0” para indisponível e “1” para disponível, o oposto do mutexes, que suportam qualquer valor inteiro.

**c. Descreva para que é utilizado a funções xSemaphoreTake() e**

**xSemaphoreGive().**

R:Ao que se refere a usabilidade pratica das funções “xSemaphoreTake()” e “xSemaphoreGive()”, no geral ambas possuem a autonomia sob o controle ao acesso de recursos compartilhados, onde a primeira tenta obter o acesso ao semáforo, se o mesmo estiver indisponível ela retornará False, onde a tarefa ficara bloqueada ate que seja liberada, se o semáforo estiver disponível a função retornara True, e a tarefa terá acesso ao recurso compartilhado. A segunda função é usada para liberar o semáforo, permitindo que outras tarefas obtenham este e acessem o recurso.

**2. Implemente um programa utilizando a estrutura semáforo no**

**FreeRTOS conforme a especificação abaixo:**

**a. Implementar um programa no FreeRTOS utilizando semáforos**

**para controlar o uso de uma variável global, está variável deve**

**simular o uso do recurso de um display;**

**b. Criar uma estrutura semáforo do tipo mutex, para tal utilizar a**

**função xSemaphoreCreateMutex();**

**c. Deverá ser criado três tarefas diferentes, todas as tarefas devem**

**escrever uma mensagem na variável display e posteriormente**

**imprimir a mensagem do display na tela;**

**d. Cada tarefa deve garantir o uso exclusivo do acesso a variável**

**display utilizando a estrutura do semáforo, para isso utilizar as**

**tarefas xSemaphoreTake() e xSemaphoreGive();**

**e. A criação das três tarefas deve ser realizada no main\_, todas as**

**tarefas criadas devem ter a mesma prioridade, passar por**

**parâmetro na criação da tarefa um número id que permite**

**identificar a tarefa;**

**f. A primeira tarefa (vTask1) deve imprimir no display a data atual**

**obtida do sistema. O código abaixo permite obter a data atual do**

**sistema, lembrando que é necessário importar a biblioteca**

**“time.h”:**

**Texto, Carta

Descrição gerada automaticamente**

**g. A segunda tarefa (vTask2) deve imprimir no display a hora atual  
obtida do sistema. O código abaixo permite obter a hora atual do  
sistema, lembrando que é necessário importar a biblioteca  
“time.h”:**

**Texto

Descrição gerada automaticamente**

**h. A terceira tarefa (vTask3) deve imprimir no display o nome da  
cidade onde você se encontra e uma temperatura obtida  
aleatoriamente. O código abaixo permite obter um número  
randômico do tipo float, a temperatura chegando até o valor  
máximo definido na constante maxTemp.**

**Texto

Descrição gerada automaticamente**