A) Analise o programa a seguir e garanta a exclusão mútua nas áreas que podem gerar condição de corrida.

```
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>
int main(){
int *soma, x=3, y=3;
int idMem; /* identificador da memória comum */
//Cria uma área de memória compartilhada
idMem = shmget(IPC_PRIVATE,sizeof(int),IPC_CREAT|0666);
if (idMem == -1) {
perror("Erro no shmget") ;
exit(0) ;
}
// acoplamento do processo a zona de memória
soma = shmat(idMem, 0, 0);
*soma = 10;
if (fork()==0){
*soma = *soma + x + y;
printf("\n\nNo filho1: soma = %d", *soma); puts(" ");
exit(0);
}
if (fork()==0){
X = X + 5;
*soma = *soma + 10;
printf("\n\nNo filho2: soma = %d", *soma); puts(" ");
exit(0);
}
wait(0); wait(0);
soma = *soma + 10;
printf("\n\nNo pai: soma = %d", *soma); puts(" ");
```

//Apaga a área de memória

shmctl(idMem, IPC_RMID, NULL);

}

B) Escreva três programas concorrentes (cada processo deve executar uma rotina/função), **tratador**, **recebedor1** e **recebedor2**, que executam um loop infinito, e que sincronizam suas ações com o uso de semáforos (operações P e V). O processo **tratador** lê valores reais (double) digitados pelo usuário, que representam a temperatura de um certo dispositivo: se a temperatura lida é menor ou igual a 25 o **recebedor1** deve ser notificado; se a temperatura lida for maior que 25 o **recebedor2** deve ser notificado. O processo **recebedor1** deverá imprimir "Temperatura baixa ou amena!" quando notificado pelo **tratador**. O processo **recebedor2** deverá imprimir "Temperatura alta!" qu notificado pelo **tratador**.

1. Represente a utilização do semáforo simbolicamente com operações P e V.
2. Implemente a solução usando semáforos no Linux.

C) Um grupo de alunos lancha em conjunto, retirando sanduíches de uma travessa que comporta até 50 biscoitos. Quando um aluno deseja comer um biscoito ele retira-o da travessa. Ao tentar pegar um biscoito, caso a travessa esteja vazia, o aluno acorda a cozinheira e espera até que esta tenha assado mais biscoitos e enchido a travessa. Considere que:

- apenas um aluno pode pegar o biscoito em um determinado momento;
- o aluno que encontrar a travessa vazia primeiro é que chama a cozinheira;
- existe apenas uma cozinheira.

Implemente três programas: para iniciar os semáforos; para representar um aluno retirando 1 biscoito; e para representar a cozinheira enchendo a travessa, que executa em loop infinito.

- podem ser executadas várias instâncias do programa que representa o aluno, inclusive, executando em paralelo, assim, a "bandeja" passa a ser uma Região crítica e deve-se usar semáforo para garantir a exclusão mútua;
- utilize semáforos para garantir a sincronização entre os processos;
- todas as esperas devem ser bloqueantes, não podendo ocorrer o problema de *Busy Waiting*.

D) Considere uma área de estacionamento com capacidade máxima para 20 carros. Enquanto há lugares livres os carros podem entrar sem problemas. No momento que o estacionamento fica cheio, os carros que chegarem devem ficar aguardando uma vaga, que é liberada por um carro que sair do estacionamento. Considerando que cada processo que executa as funções é um carro, faça três programas em C, iniciaSemaforo.c, entraCarro.c e saiCarro.c, representando o controle de acesso com a utilização de semáforos.

- Quando o semáforo é inicializado apresenta a mensagem "semáforo inicializado".
- Quando o programa entraCarro é executado apresente a mensagem "O carro **<pid** do processo> chegou!"e quando este consegue entrar no estacionamento indique essa situação com a mensagem "O carro **<pid** do processo> estacionou!".
- Da mesma forma, quando o programa saiCarro é executado, libera uma vaga no estacionamento. Indique essa situação com a mensagem "O carro **<pid** do processo> saiu!".