8)

1. Como a região crítica é um local onde os processos podem entrar para poder usar os recursos compartilhados disponíveis, cuidados devem ser tomados para que não ocorra condições de corrida, ou seja, que os processos tentem acessar os mesmos recursos ao mesmo tempo. Para isso o semáforo é uma estratégia ideal, pois ao definirmos um valor inicial para o mesmo, podemos dizer quantos processos podem acessar essa região crítica nesse momento. Devemos iniciar o semáforo com tamanho 1. Ao fazermos isso, podemos dizer que uma thread pode utilizar os recursos que são compartilhados, porém de acordo com a implementação dada, temos a chamada do método wait(S) que irá decrementar o valor do semáforo, ou seja, apenas uma thread vai ter acesso a essa região e o valor desse semáforo vai para zero, fazendo com que esse acesso seja protegido com exclusão mútua, isso me garante que enquanto o recurso estiver em uso, qualquer outro processo que o queira utilizar deve esperar a liberação. Logo em seguida vemos que existe uma chamada para o método post(S) que irá incrementar um valor no semáforo, fazendo com que esse volte novamente para um, porém isso já não é um problema, visto que a próxima região é não-crítica, ou seja, sem possibilidades de existir condições de corrida.
2. A partir do momento que o semáforo for inicializado com tamanho 13, assim que ocorrer a chamada do método wait(S) esse semáforo irá ser decrementado e irá para o tamanho 12, fazendo com que 12 threads possam acessar concorrentemente à região crítica e a que já está dentro dela. Assim nós podemos ter muitos problemas de condições de corrida, ou seja, uma thread pode me retornar os valores errados, por causa da ocorrência dos outros 12 eventos que podem estar ocorrendo ao mesmo tempo.
   1. Se os dois semáforos (Q e S) começarem com o valor 1, acontecerá que os processos poderão acessar as regiões críticas em semáforos diferentes simultaneamente, pois assim que entrar nos respectivos semáforos, o wait() irá fazer com que eles sejam zerados e que nenhum outro processo tente entrar enquanto o processo atual esteja utilizando os recursos da região crítica. Assim que o processo termina de utilizar, irá ser chamado o post() que irá acrescentar um valor ao semáforo, fazendo com que esses estejam disponíveis novamente.
   2. O processo 1 será executado porém o processo 2 não. Assim que o processo 1 verificar que o semáforo está com valor 1, ele entrará na região crítica, porém quando o processo 2 verificar que o semáforo está com o valor de zero, esse processo 2 irá dormir e esperar que outro processo o acorde quando for realizada a chamada do post(), ou seja, irá acontecer a utilização dos semáforos de forma exclusiva, primeiro o processo 1 no semáforo Q e depois que esse acabar o processo 2 no semáforo S.
   3. Os valores dos semáforos serem iniciados com zero, faz com que os processos, tanto o 1 quanto o 2, sejam colocados em para dormir e não sejam executados em nenhum momento, pois não vai existir outro processo que o acorde.
3. Dado o código podemos afirmar que os dois semáforos começando com o valor de um, ocorrerá um deadlock, pois quandos os dois processos entrarem na região crítica e chamarem o wait() não irá existir nenhum outro processo para os liberarem da situação que estão, ou seja, os dois semáforos ficaram zerados e não poderão mais ser usados para os processos acessarem a região crítica.