Assignment 2

1)

On a system using Round Robin Scheduling, let s represent the time required to perform a process switch, q represent the RR time quantum, and r represent the average time a process runs before blocking on I/O. Give a formula for CPU efficiency given the following:

i) $q=\infty$ ii) $q>r$ iii) $s<q<r$ iv) $s=q<r$ v) q nearly 0

2) Write a procedure to synchronize the situation using Semaphores:

A small bridge connects the 2 banks of a river. The bridge is so narrow that at any time only 1 person can cross through the bridge. A person on the other side must wait till the person from one side has crossed completely.

s = time required to perform a process switch
q = quantum time for Round Robin scheduling
r = avg. time a process runs before blocking on I/O

$$\text{CPU efficiency} = \frac{\text{Useful cpu time}}{\text{Total cpu time}} \times 100\%$$

i) $q = \infty$

Here quantum is so large that every process completes before quantum is over. Hence round robin transformes into FCFS and no switching is required.

$$\therefore \text{CPU efficiency} = \frac{r}{r} \times 100\% = 100\%$$

ii) $q > r$

Here also process completes before quantum getting over.

$$\therefore \text{CPU efficiency} = \frac{r}{r} \times 100\% = 100\%$$

iii) $s < q < r$

Here $\frac{r}{q}$ number of switches are required for the process complete, thus total switching time $= s \times \left(\frac{r}{q}\right)$

$$\therefore \text{CPU efficiency} = \frac{r}{r + \left(\frac{sr}{q}\right)} = \frac{qr}{qr + sr} = \frac{q}{q+s}$$

iv) $s = q < r$

Here also similar to (iii) ha[illegible] and putting $s = q$

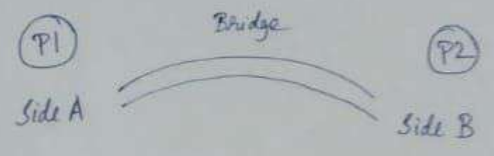
$$\therefore \text{CPU efficiency} = \frac{q}{2q} = [illegible] 50\%$$

v) q nearly 0

Here as q is so small [illegible] rather than executing the p[illegible]
$q \to 0$ in (iii)

$$\therefore \text{CPU efficiency} = 0$$

Q2>



Semaphore
$s = 1$

Pseudocode :

Process P1 :

wait (s)
[Person walks from side A to side B on the bridge]
signal (s)
[Person walks around in side A]

Process P2 :

wait (s)
[Person walks from side B to side A on the bridge.]
signal (s)
[Person walks around in side B]

```
wait (Semaphore s) {
      while (s == 0)   ;
      s = s - 1 ;
}
```

```
signal (Semaphore s) {
      s = s + 1 ;
}
```

By using semaphore s , ~~to~~ mutually exclusive processes P1 and P2 are executed synchronously over the shared resource which is "bridge" here.