Explique **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:
I) *(1 p)* La estrategia de *prevención de interbloqueos* consiste en conceder a un proceso solamente aquellas
peticiones de recursos que tengan garantizado que no conducirán a un estado de interbloqueo.

Verdadero. Se prevé todos los recursos que va a usar el proceso para su ejecución , si requiere mas recursos de lo que están permitidos no será adminito para ser ejecutado. Y mientra que se esta ejecutando el proceso va solicitando los recursos, si
la asignación de uno de estos recursos con lleva al interbloqueo, este proceso pasa al estado bloqueado y espera hasta que pueda usarlo.

II) *(1 p)* La planificación expropiativa produce un menor sobrecarga al sistema que una planificación no expropiativa.

Verdadero. Produce una menor sobrecarga del sistema, pero también disminuye el rendimiento del mismo.
La expropiativa, aumenta la sobrecarga pero también aumenta el rendimiento del sistema.

III) *(1p)*Una de las principales ventajas que tiene implementar una determinada aplicación como uno o varios procesos multihilos es que permite aumentar el rendimiento del sistema.

Verdadero. Entre otras ventajas, esta El ahorro de los recursos, un mejor aprovechamiento de la arquitectura del microprocesador, una mejor comunicación entre procesos, simplificación de la estructura de las aplicaciones.

IV) *(1 p)* Un sistema operativo multiacceso debe ser capaz de soportar necesariamente multiprogramación.

Falso. No requiere necesariamente la multiprogramación, permite el acceso al sistema informático a través de dos o mas terminales.

2. *(2 p)* Supóngase un sistema que permite retención y espera, exclusión mutua y expropiación en sus recursos. En un determinado instante de tiempo el grafo de asignación de los recursos R1, R2 y R3 del sistema a los procesos A, B y C es el que se muestra en la figura. Explicar **razonadamente** si se cumplen las condiciones para que se produzca una situación de *interbloqueo*.

En esta caso tenemos los procesos A,B,C que están usando los recursos R1, R2, R3, respectivamente, el proceso A esta usando el recurso R1, pero quiere usar el proceso R2, que lo esta usando B, y lo mismo pasa con el proceso C, un proceso solo puede cambiar de recurso si el recurso de destino esta libre, como en este casi ninguno de los recursos de

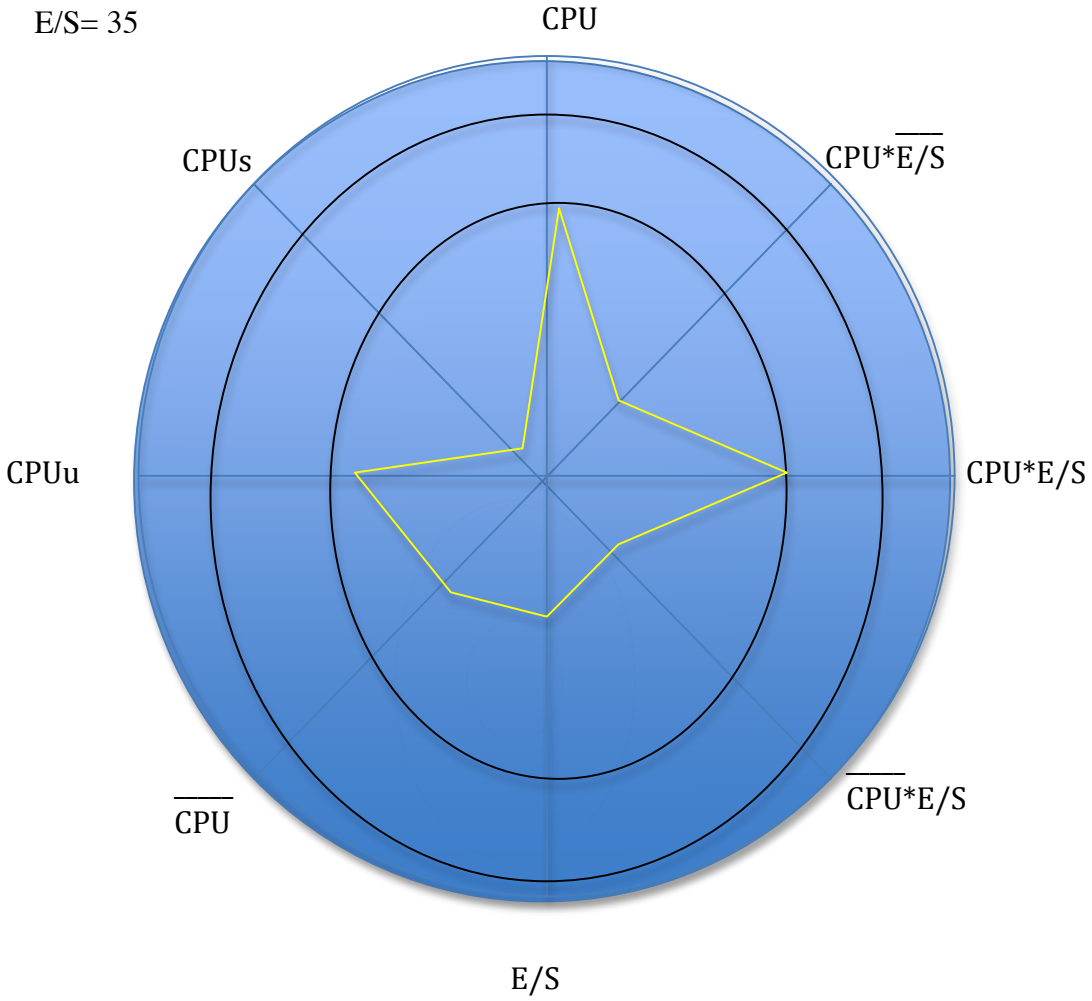
destino esta libre, ningún proceso puede abandonar el recurso que esta usando, y se bloquean entre ellos.

(2 p) Durante un cierto intervalo de observación de 100 ut, la CPU y los dispositivos de E/S de un computador han sido utilizados por el sistema operativo y varios procesos de usuarios de acuerdo con el diagrama de uso que se muestra en la figura. Dibujar y comentar el diagrama de Kiviatt - Kent asociado a este sistema informático. Suponer que el tiempo de ejecución en modo supervisor ha sido el 12 % del tiempo total de uso de la CPU.

CPU: Uso de la CPU 60 u.t
CPUs: El tiempo de la Cpu es de 12 u.t en modo supervisor
CPUu: 60 – 12 = 48, en modo usuario.

$\overline{CPU}$= 40 u.t
CPU*E/S= 40 u.t

CPU*$\overline{E/S}$= 17 u.t
E/S= 35



Problema 4.

Explicacion.

Usare dos semáforos para coordinar los dos procesos, en el momento que el proceso A entre en tareas 1, pongo el semáforo del proceso A a 1, y lo mismo pasa con el proceso B, en el momento que el proceso A, entre para ejecutar la tarea 1 y haya puesto su semáforo a 1, comprueba si B ha hecho lo mismo, y si los dos procesos tienen los semáforos a 1, empieza la ejecución, una vez terminada la ejecución, ambos procesos ponen sus semáforos a 0, para poder realizar la siguiente tarea.

```
Semáforo_binario S1,S2 ;

Void proceso_A ( )

 if signal_sem(s1)==0
{

signal_sem(s1)

while  (signal_sem(s1)==1 )

{

if  (singal_sem(s2)==1)
Tareas1
wait_sem(s1)
}

if signal_sem(s1)==0
{
signal_sem(s1)
while  (signal_sem(s1)==1 )

{

if  (singal_sem(s2)==1)


Tareas2

wait_sem(s1)
}
```

```
}

Void proceso_B ( )


 if signal_sem(s2)==0
{

signal_sem(s2)

while  (signal_sem(s2)==1 )

{

if  (singal_sem(s1)==1)


Tareas3

wait_sem(s2)


}

if signal_sem(s2)==0
{

signal_sem(s2)

while  (signal_sem(s2)==1 )

{

if  (singal_sem(s1)==1)


Tareas4

wait_sem(s2)
}
}
```

```
void main ()
{
init_sem(s1,1)
init_sem(s2,1)

ejecución_concurrente ( procesoA, procesoB)
}
```