Carlos Rafael Flores Gallardo is727635
Carlos Rogelio Quirarte Vázquez is724644

Actividad 11

1.

P(0)	P(1)	flag[0]	flag[1]	turno
		FALSE	FALSE	0
flag[0]=TRUE;		TRUE	FALSE	0
	flag[1]=TRUE;	TRUE	TRUE	0
while(turno != 0)		TRUE	TRUE	0
	while(turno != 1)	TRUE	TRUE	0
CS		TRUE	TRUE	0
	while(flag[0])	TRUE	TRUE	0
flag[0] = FALSE;		FALSE	TRUE	0
	turno = 1;	FALSE	TRUE	1
flag[0] = TRUE;		TRUE	TRUE	1
	while(flag[0])	TRUE	TRUE	1
while(turno != 0)		TRUE	TRUE	1
	turno = 1;	TRUE	TRUE	1
while(flag[1])		TRUE	TRUE	1
	while(flag[0])	TRUE	TRUE	1
turno = 0;		TRUE	TRUE	0
	turno = 1;	TRUE	TRUE	1
while(flag[1])		TRUE	TRUE	1
	while(flag[0])	TRUE	TRUE	1
while(flag[1])		TRUE	TRUE	1
	while(flag[0])	TRUE	TRUE	1

Ambos procesos pusieron su flag en TRUE, por lo que ambos ciclos while se quedaran repitiéndose infinitamente.

Carlos Rafael Flores Gallardo is727635
Carlos Rogelio Quirarte Vázquez is724644

2.

P(0)	P(1)	P(2)	flag[0]	flag[1]	flag[2]	turno
			FALSE	FALSE	FALSE	0
flag[0]=TRUE;			TRUE	FALSE	FALSE	0
			TRUE	FALSE	FALSE	0
		flag[2]=TRUE;	TRUE	FALSE	TRUE	0
while(flag[1] || flag[2])			TRUE	FALSE	TRUE	0
			TRUE	FALSE	TRUE	0
		while(flag[0] || flag[1])	TRUE	FALSE	TRUE	0
flag[0] = FALSE			FALSE	FALSE	TRUE	0
			FALSE	FALSE	TRUE	0
		flag[2] = FALSE	FALSE	FALSE	FALSE	0
while(turno!=0)			FALSE	FALSE	FALSE	0
			FALSE	FALSE	FALSE	0
		while(turno!=2)	FALSE	FALSE	FALSE	0
flag[0] = TRUE;			TRUE	FALSE	FALSE	0
			TRUE	FALSE	FALSE	0
		while(turno != 2)	TRUE	FALSE	FALSE	0
while(flag[1] || flag[2])			TRUE	FALSE	FALSE	0
			TRUE	FALSE	FALSE	0
		while(turno != 2)	TRUE	FALSE	FALSE	0
CS			TRUE	FALSE	FALSE	0
			TRUE	FALSE	FALSE	0
		while(turno != 2)	TRUE	FALSE	FALSE	0
turno = 1;			TRUE	FALSE	FALSE	1
			TRUE	FALSE	FALSE	1
		while(turno != 2)	TRUE	FALSE	FALSE	1
flag[0] = FALSE			FALSE	FALSE	FALSE	1
			FALSE	FALSE	FALSE	1
		while(turno != 2)	FALSE	FALSE	FALSE	1
RS			FALSE	FALSE	FALSE	1
			FALSE	FALSE	FALSE	1
		while(turno != 2)	FALSE	FALSE	FALSE	1
flag[0] = TRUE;			TRUE	FALSE	FALSE	1
			TRUE	FALSE	FALSE	1
		while(turno != 2)	TRUE	FALSE	FALSE	1
while(flag[1] || flag[2])			TRUE	FALSE	FALSE	1
			TRUE	FALSE	FALSE	1
		while(turno != 2)	TRUE	FALSE	FALSE	1
CS			TRUE	FALSE	FALSE	1
			TRUE	FALSE	FALSE	1
		while(turno != 2)	TRUE	FALSE	FALSE	1
turno = 1;			TRUE	FALSE	FALSE	1

Si nunca si inicializa el proceso 1, se terminará ciclando el turno, como se ve en la línea amarilla, y esto se atorará ahí.

3.

P(0)	P(1)	P(2)	flag[0]	flag[1]	flag[2]	turno
			FALSE	FALSE	FALSE	0
flag[0] = TRUE;			TRUE	FALSE	FALSE	0
	flag[1] = TRUE;		TRUE	TRUE	FALSE	0
			TRUE	TRUE	FALSE	0
turno = 1			TRUE	TRUE	FALSE	1
	turno = 2		TRUE	TRUE	FALSE	2
			TRUE	TRUE	FALSE	2
while((flag[1] || flag[2]) && turno!=0)			TRUE	TRUE	FALSE	2
	while((flag[2] || flag[0]) && turno!=1)		TRUE	TRUE	FALSE	2
			TRUE	TRUE	FALSE	2
while((flag[1] || flag[2]) && turno!=0)			TRUE	TRUE	FALSE	2
	while((flag[2] || flag[0]) && turno!=1)		TRUE	TRUE	FALSE	2

Si nunca se inicia el proceso 2, se quedará esperando para siempre a que ese inicie, lo cual nunca pasara, y se quedara en la línea amarilla permanentemente.

Conclusiones.

- **Carlos Flores:** Pude observar que a pesar de que estas soluciones son mucho mas optimas que lo que habíamos visto en los anteriores intentos, pero a pesar de esto, hay casos especiales los cuales harán que estos algoritmos mal funcionen, sobre todo cuando se usan mas de 2 procesos.
- **Carlos Rogelio Quirarte Vázquez:** Los procesos necesitan algoritmos para sincronizarse, durante esta práctica pudimos poner a prueba ciertos algoritmos o intentos de sincronización que pudieran verse como perfectamente funcionales y eficientes sobre el papel, pero poniéndolos a prueba podemos ver cómo en ciertos casos es posible que fallen de manera catastrófica, por lo que, aunque la idea parezca buena. Siempre debemos de hacer muchas pruebas antes de considerar que nuestro algoritmo es correcto, y así evitar llevarlo más lejos en su desarrollo, sólo me queda la duda sobre cómo es que funcionarán los semáforos de los sistemas operativos que usamos hoy en día