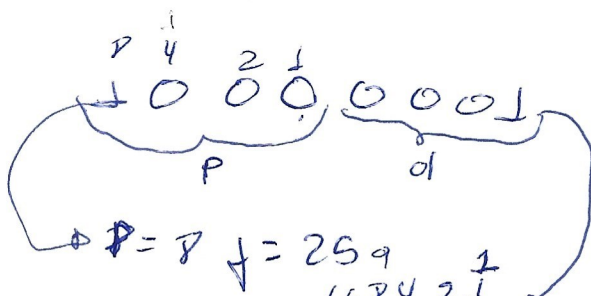


1- a) O campo P possui 16 linhas $2^4 = 16$ portanto o campo P possui 4 bits

b) O maior número no campo f é $31_{10} = 11111_2$, portanto f possui 5 bits

c) 129 →

128	64	32	16	8	4	2	1
1	0	0	0	0	0	0	1



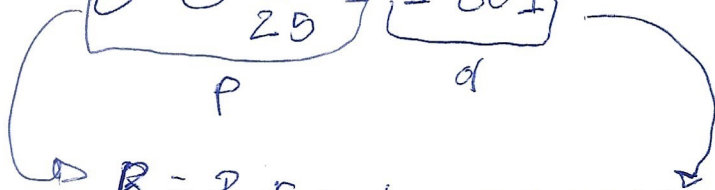
168421
100010001
9132168421

40

10010001
256 128 64 32 16 8 4 2 1

57 →

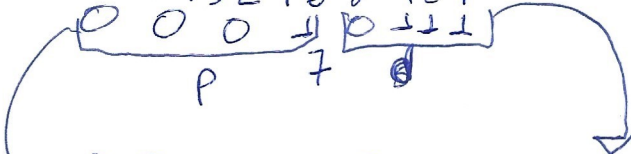
128	64	32	16	8	4	2	1
0	0	1	1	0	0	1	0



$P = 3 \quad f = 1 = 000011001 = 25$
168421

23 →

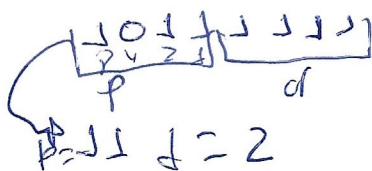
128	64	32	16	8	4	2	1
0	0	0	1	0	1	1	1



$P = 1 \quad f = 7 = 001110111 = 119$
6432168421

191 →

128	64	32	16	8	4	2	1
1	0	1	1	1	1	1	1

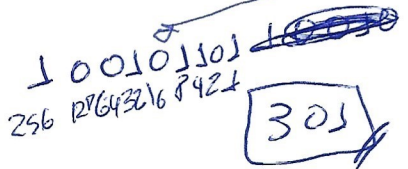


$P = 1 \quad f = 2$

00010111 → 47
72168421

93 →

128	64	32	16	8	4	2	1
0	1	0	1	1	0	1	0



$P = 5 \quad f = 48$
100101101
256 128 64 32 16 8 4 2 1

30

137 →

128	64	32	16	8	4	2	1
1	0	0	0	1	0	0	1



$P = 8 \quad f = 25$

110011001
256 128 64 32 16 8 4 2 1

409

29 →

128	64	32	16	8	4	2	1
0	0	0	1	1	0	1	1



$P = 1 \quad f = 7$

00111011
64 32 16 8 4 2 1

125

12 →

128	64	32	16	8	4	2	1
0	0	0	0	1	1	0	0

$P = 0 \quad f = 23$

10111100
256 128 64 32 16 8 4 2 1

380

46 - 128 64 32 16 8 4 2 1
 0 0 1 0 1 1 0

$p = 2, d = 0$

00000 1110
 128 64 32 16 8 4 2 1
 14

150 - 128 64 32 16 8 4 2 1
 1 0 0 1 0 1 0

$p = 9, d = 14 \rightarrow 11100110$
 128 64 32 16 8 4 2 1
 230

20 - 128 64 32 16 8 4 2 1
 0 0 0 1 0 1 0 0

$p = 1, d = 7$
 001110100
 64 32 16 8 4 2 1
 116

a) as tabelas nível 1 e nível 2 possuem 4 linhas, $2^2 = 4$, portanto p_1 e p_2 possuem 2 bits

b) o maior número do campo é 15₁₀, 1111₂, portanto possui 4 bits

c) 27 \rightarrow 256 128 64 32 16 8 4 2 1
 0 0 0 0 1 1 1 1
 $p_1 = 0, p_2 = 1$
 $f = 101111011$
 256 128 64 32 16 8 4 2 1
 439

202
 $p_1 = 3, p_2 = 0$
 $f = 11001010$
 256 128 64 32 16 8 4 2 1
 250

190
 $p_1 = 2, p_2 = 3$
 $f = 200101110$
 32 16 8 4 2 1
 196

15 - 0 0 0 0 1 1 1 1
 $p_1 = 0, p_2 = 0$
 $f = 111011111$
 128 64 32 16 8 4 2 1
 191

116 - 0 1 1 0 1 0 0 1
 $p_1 = 1, p_2 = 3$
 $f = 501010100$
 64 32 16 8 4 2 1
 84

162 - 1 0 1 0 0 0 1 0
 $p_1 = 2, p_2 = 2$
 $f = 101010010$
 128 64 32 16 8 4 2 1
 162

$$221 \ 29 \quad \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 1 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 1 & 0 & 1 & 0 \\ \hline \end{array}$$

$P_1 \quad P_2$

$$P_1 = 0 \\ P_2 = 1 \quad f = 8 - \begin{array}{|c|c|c|c|} \hline 1 & 0 & 0 & 0 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 1 & 0 & 1 & 0 \\ \hline \end{array} = 145$$

$128 \ 64 \ 32 \ 16 \ 8 \ 4 \ 2 \ 1$

$$12 - \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 9 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 1 & 1 & 0 & 0 \\ \hline \end{array}$$

$P_1 \quad P_2 \quad d$

$$P_1 = 0 \\ P_2 = 0 \quad f = 11 = \begin{array}{|c|c|c|c|} \hline 1 & 0 & 1 & 1 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 1 & 1 & 0 & 0 \\ \hline \end{array}$$

$128 \ 64 \ 32 \ 16 \ 8 \ 4 \ 2 \ 1$

$$47 - \begin{array}{|c|c|c|c|} \hline 0 & 0 & 1 & 0 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 1 \\ \hline \end{array}$$

$P_1 \quad P_2 \quad d$

$$P_1 = 0 \\ P_2 = 2 \quad f = 4 = \begin{array}{|c|c|c|c|} \hline 0 & 1 & 0 & 0 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 1 \\ \hline \end{array}$$

$64 \ 32 \ 16 \ 8 \ 4 \ 2 \ 1$

$$5 - \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 0 & 1 & 1 & 1 \\ \hline \end{array}$$

$P_1 \quad P_2 \quad d$

$$P_1 = 0 \\ P_2 = 0 \quad f = 11 = \begin{array}{|c|c|c|c|} \hline 1 & 0 & 1 & 1 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 0 & 1 & 1 & 1 \\ \hline \end{array}$$

$128 \ 64 \ 32 \ 16 \ 8 \ 4 \ 2 \ 1$

$$132 - \begin{array}{|c|c|c|c|} \hline 1 & 0 & 0 & 0 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 0 & 0 & 1 & 0 \\ \hline \end{array}$$

$P_1 \quad P_2 \quad d$

$$P_1 = 2 \\ P_2 = 0 \quad f = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 0 & 1 & 0 & 0 \\ \hline \end{array}$$

$128 \ 64 \ 32 \ 16 \ 8 \ 4 \ 2 \ 1$

$$f = 9 \quad 10010100 = 148$$

3-a) ~~Para~~ Para 4 procesos, $z=4$
 Portanto $PID = 2$, ~~processos~~ ^{processos} para 16
 páginas $2^4 = 16$ portanto para 4
 bits para P e cada página para 5
 endereços então $2^3 = 8$, 0 para 3 bits.

b) O maior número é 63 então
 $f = 6$ bits, sendo assim, sim é
 possível calcular.

$$c) 431 - \begin{array}{|c|c|c|c|} \hline 1 & 0 & 1 & 0 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 1 \\ \hline \end{array}$$

$PID \quad P \quad d$

$$PID \ P = 110101 = 53$$

$$f = 32 = 100000111$$

$$510 - \begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 1 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 1 & 1 & 0 & 0 \\ \hline \end{array}$$

$PID \quad P \quad d$

$$PID \ P = 111111 = 63$$

$$f = 59 = 111011110$$

$$152 - \begin{array}{|c|c|c|c|} \hline 0 & 1 & 0 & 0 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 1 & 1 & 0 & 0 \\ \hline \end{array}$$

$PID \quad P$

$$PID \ P = 010011 = 19$$

$$f = 57 = 111001000$$

$$235 - \begin{array}{|c|c|c|c|} \hline 0 & 1 & 1 & 0 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 1 & 0 & 1 & 1 \\ \hline \end{array}$$

$PID \quad P \quad d$

$$PID \ P = 011101 = 29$$

$$f = 10 = 001010011$$

$$315 - \begin{array}{|c|c|c|c|} \hline 1 & 1 & 0 & 1 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 1 & 1 & 0 & 1 \\ \hline \end{array}$$

$PID \quad P \quad d$

$$PID \ P = 110111 = 55$$

$$f = 29 = 11101011$$

$$92 - \begin{array}{|c|c|c|c|} \hline 0 & 0 & 1 & 0 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 1 & 0 & 1 & 1 \\ \hline \end{array}$$

$PID \quad P \quad d$

$$PID \ P = 001011 = 11$$

$$f = 53 = 110101100$$

$$2 - \begin{array}{|c|c|c|c|} \hline 0 & 0 & 0 & 0 \\ \hline \end{array} \begin{array}{|c|c|c|c|} \hline 0 & 1 & 0 & 0 \\ \hline \end{array}$$

$PID \quad P \quad d$

$$PID \ P = 000000 = 0$$

$$f = 11 = 101000010$$

51 - $\underbrace{100}_{pid} \underbrace{0110}_P \underbrace{1011}_d$

$\rightarrow pid P = 000110 \rightarrow P6$

$d = 50 \quad 100010011 \quad \boxed{403}$

389 $\underbrace{11}_{pid} \underbrace{0000}_P \underbrace{100}_d$

$\rightarrow pid P = 110000 = 48$

$d = 39 \quad 100111101 \quad \boxed{317}$

2. DEADLOCK

- As tabelas a seguir apresentam as matrizes alocação, máximo e o vetor disponível para um conjunto de processos/recursos em um dado sistema operacional. Para cada um dos cenários, verifique se o sistema está ou não em deadlock. Em caso de não deadlock, apresente uma sequência de execução acompanhada do valor do vetor disponível após a execução de cada processo. Em caso de deadlock, justifique sua resposta, apresentando a matriz necessária.

A)	Disponível						
	A			B		C	
	1		2		1		
	Alocação			Máximo			
	A	B	C	A	B		C
P ₀	2	2	3	5	4		3
P ₁	3	1	0	7	2		2
P ₂	1	2	0	3	3		1
P ₃	0	1	1	2	4		2
P ₄	4	1	0	4	2		0

B)	Disponível						
	A			B		C	
	1		1		2		
	Alocação			Máximo			
	A	B	C	A	B		C
P ₀	1	2	1	4	3		1
P ₁	2	3	1	5	3		2
P ₂	1	3	1	2	4		6
P ₃	1	0	0	3	4		1
P ₄	1	2	2	5	3		4

C)	Disponível						
	A			B		C	
	0		1		3		
	Alocação			Máximo			
	A	B	C	A	B		C
P ₀	1	5	0	4	4		2
P ₁	1	0	3	5	0		5
P ₂	1	1	0	2	2		1
P ₃	1	0	2	3	0		4
P ₄	1	1	1	5	4		5

A) Sem deadlock, todos os processos foram executados.

	Alocação			Máximo			Necessário			Disponível			
	A	B	C	A	B	C	A	B	C	A	B	C	EXECUTOU
P0	2	2	3	5	4	3	3	2	0	1	2	1	P4
P1	3	1	0	7	2	2	4	1	2	5	3	1	P2
P2	1	2	0	3	3	1	2	1	1	6	5	1	P3
P3	0	1	1	2	4	2	2	3	1	6	6	2	P1
P4	4	1	0	4	2	0	0	1	0	9	7	2	P0
										11	9	5	

B) Existe DeadLock logo no início, nenhum processo pode ser executado.

[illegible]

C) Existe DeadLock logo no início, nenhum processo pode ser executado.

[illegible]

D)	Disponível					
	A		B		C	
	2		2		3	
	Alocação			Máximo		
	A	B	C	A	B	C
P ₀	2	2	3	2	2	3
P ₁	3	1	0	5	1	2
P ₂	1	2	0	3	3	1
P ₃	2	1	1	2	3	2
P ₄	4	1	0	4	2	0

E)	Disponível					
	A		B		C	
	0		4		2	
	Alocação			Máximo		
	A	B	C	A	B	C
P ₀	4	2	1	4	4	2
P ₁	2	3	1	6	3	3
P ₂	2	3	1	2	4	6
P ₃	1	0	0	2	3	1
P ₄	1	2	2	5	3	4

F)	Disponível					
	A		B		C	
	3		1		0	
	Alocação			Máximo		
	A	B	C	A	B	C
P ₀	0	4	0	3	4	2
P ₁	1	0	3	2	2	5
P ₂	1	1	0	3	1	1
P ₃	1	0	2	1	0	4
P ₄	1	1	1	4	2	5

D) Sem deadlock, todos os processos foram executados.

	Alocação			Máximo			Necessário			Disponível			
	A	B	C	A	B	C	A	B	C	A	B	C	EXECUTOU
P ₀	2	2	3	2	2	3	0	0	0	2	2	3	P ₀
P ₁	3	1	0	5	1	2	2	0	2	4	4	6	P ₁
P ₂	1	2	0	3	3	1	2	1	1	7	5	6	P ₂
P ₃	2	1	1	2	3	2	0	2	1	8	7	6	P ₃
P ₄	4	1	0	4	2	0	0	1	0	10	8	7	P ₄
										14	9	7	

E) Sem deadlock, todos os processos foram executados.

	Alocação			Máximo			Necessário			Disponível			
	A	B	C	A	B	C	A	B	C	A	B	C	EXECUTOU
P ₀	4	2	1	4	4	2	0	2	1	0	4	2	P ₀
P ₁	2	3	1	6	3	3	4	0	2	4	6	3	P ₁
P ₂	2	3	1	2	4	6	0	1	5	6	9	4	P ₃
P ₃	1	0	0	2	3	1	1	3	1	7	9	4	P ₄
P ₄	1	2	2	5	3	4	4	1	2	8	11	6	P ₂

	Alocação			Máximo			Necessário			Disponível			
										10	14	7	

F) Existe DeadLock logo no início, nenhum processo pode ser executado.

	Alocação			Máximo			Necessário			Disponível			
	A	B	C	A	B	C	A	B	C	A	B	C	EXECUTOU
P0	0	4	0	3	4	2	3	0	2	3	1	0	
P1	1	0	3	2	2	5	1	2	2				
P2	1	1	0	3	1	1	2	0	1				
P3	1	0	2	1	0	4	0	0	2				
P4	1	1	1	4	2	5	3	1	4				

G)	Disponível						
	A			B		C	
	2			5		0	
	Alocação			Máximo			
	A	B	C	A	B		C
P ₀	2	2	3	4	4		3
P ₁	4	1	0	7	1		0
P ₂	1	2	3	3	3		3
P ₃	2	1	1	2	4		2
P ₄	4	1	1	4	2		0

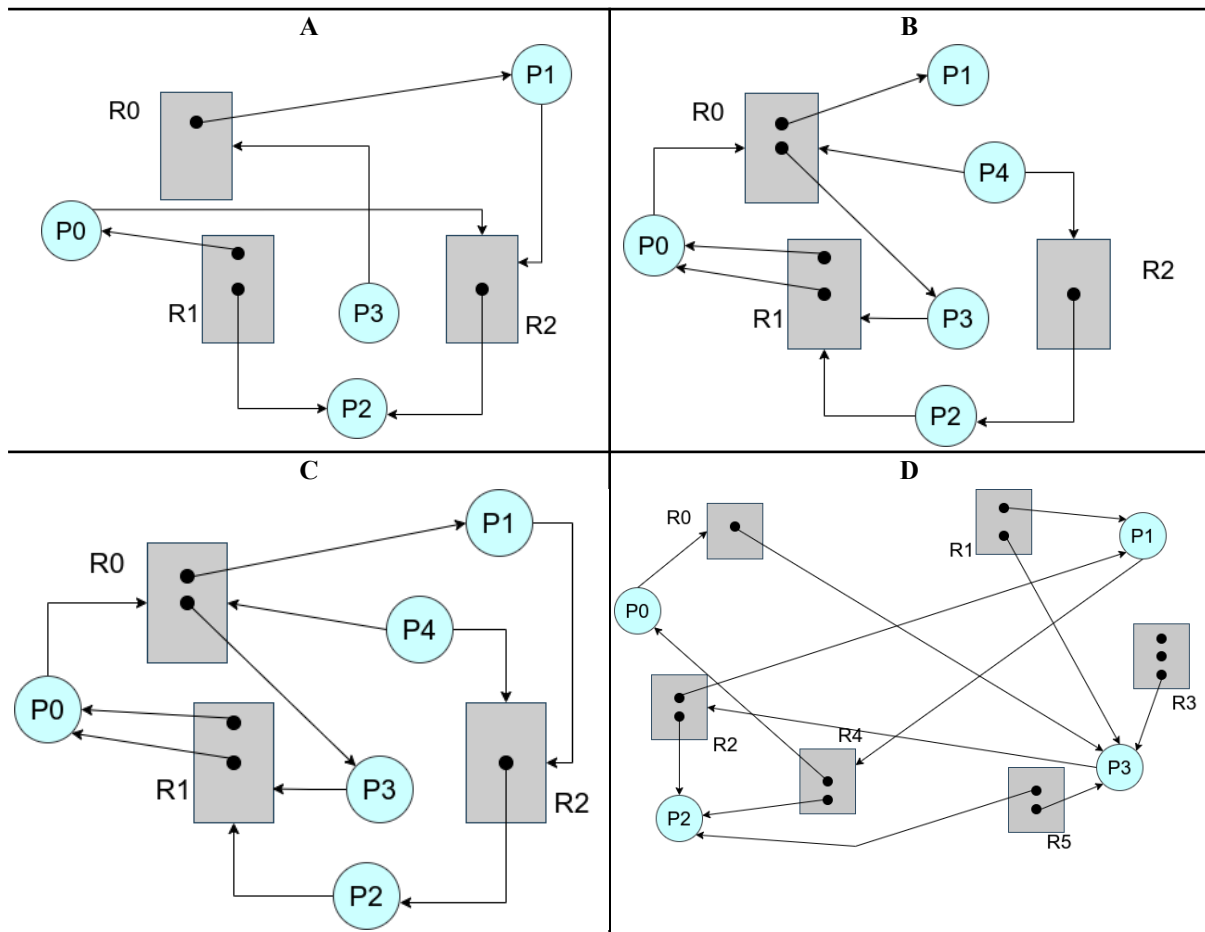
H)	Disponível						
	A			B		C	
	1			1		1	
	Alocação			Máximo			
	A	B	C	A	B		C
P ₀	3	5	1	4	6		1
P ₁	2	3	1	5	3		2
P ₂	1	3	5	2	4		6
P ₃	2	3	0	3	4		1
P ₄	1	2	2	5	3		4

I)	Disponível						
	A			B		C	
	0			0		3	
	Alocação			Máximo			
	A	B	C	A	B		C
P ₀	1	4	0	3	4		2
P ₁	1	2	3	2	0		5
P ₂	3	2	0	3	2		1
P ₃	1	0	3	1	0		4
P ₄	3	5	3	5	5		5

G) Erro no processo 4, alocação maior que o máximo (P4-C).

	Alocação			Máximo			Necessário			Disponível			
	A	B	C	A	B	C	A	B	C	A	B	C	EXECUTOU
P0	2	2	3	4	4	3	2	2	0	2	5	0	
P1	4	1	0	7	1	0	3	0	0				
P2	1	2	3	3	3	3	2	1	0				

1. Analise os seguintes grafos de alocação de recursos. Há presença de deadlock? Justifique sua resposta:



A) Não possui deadlock.

P3 está bloqueado pois não possui R0 que está sendo usado por P1.

P1 está bloqueado pois não possui R2 que está sendo usado por P2.

P0 está bloqueado pois não possui R2 que está sendo usado por P2.

P2 não está bloqueado, pois possui todos os recursos que precisa R2 e R1.

P2 executa -> libera R2 e R2

P2 executa -> Desbloqueia P1 e P0

P1 executa -> Desbloqueia P3

B) Não tem deadlock

P0 está bloqueado pois não possui R0 que está sendo usado por P1 e P3

P2 está bloqueado pois não possui R1 que está sendo usado por P0

P3 está bloqueado pois não possui R1 que está sendo usado por P0

P4 está bloqueado pois não possui R2 que está sendo usado por P2

P1 não está bloqueado pois possui todos os recursos que precisa R0

P1 executa -> Libera R0 e Desbloqueia P0

P0 executa -> Libera R1

P0 executa -> Desbloqueia P3 e P2

P2 executa -> Libera R2 e Desbloqueia P4

C) Tem deadlock.

P0 está bloqueado pois não possui R0 que está sendo usado por P3 e P1

P1 está bloqueado pois não possui R2 que está sendo usado por P2

P2 está bloqueado pois não possui R1 que está sendo usado por P0

P3 está bloqueado pois não possui R1 que está sendo usado por P0

P4 está bloqueado pois não possui R2 que está sendo usado por P2

D) Não possui deadlock.

P0 está bloqueado pois não possui R0 que está sendo usado por P3

P3 está bloqueado pois não possui R2 que está sendo usado por P2 e P1

P1 está bloqueado pois não possui R4 que está sendo usado por P0 e P2

P2 não está bloqueado. Ele possui todos os recursos que precisa: R2, R4 e R5

P2 executa -> Libera R4 e R2

P2 executa -> Desbloqueia P1 e P3

P3 executa -> Libera R0

P3 executa -> Desbloqueia P0

3. SINCRONIZAÇÃO

1) Operações com semáforos

A seguir, apresentamos uma sequência de operações do semáforo no início e no final das tarefas A, B, C. Considere que cada tarefa executa em um núcleo de processador dedicado. E considere que cada ação (P(Sx), V(Sx) ou .) possui tempo igual a 1T.

	Task A	Task B	Task C
1	P(SA)	P(SB)	P(SC)
2	P(SA)	.	P(SC)
3	P(SA)	.	P(SC)
4	.	.	.
5	.	.	.
6	.	V(SC)	V(SB)
7	V(SB)	V(SA)	V(SB)
8	END	.	V(SA)
9		END	END

Determine para os 6 casos a,b,c,d,e,f apresentados na tabela abaixo, se e em qual sequência as tarefas são executadas, usando as inicializações das variáveis do semáforo dadas na tabela.

Semáforos	a)	b)	c)	d)	e)	f)	g)
SA	2	3	2	0	3	1	1
SB	0	0	1	0	1	0	1
SC	2	2	1	3	3	3	1

A) Deadlock, nenhuma task finaliza, TA e TC bloqueadas no T3 (tempo 3) e TB bloqueado em T1 (tempo 1).

Semáforos	a)
SA	2 1 0
SB	0
SC	2 1 0

Task	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T3
TA	P(SA)	P(SA)	X										
TB	X	X	X										
TC	P(SC)	P(SC)	X										

B) Não existe Deadlock, Todas as tasks finalizam. TA em T8(tempo 8), TB em T16(tempo 16) e TC em T20 (tempo 20). Com isso concluímos que qualquer teste que tenha no mínimo SA, SB e SC como 3,0 e 2 respectivamente, não haverá deadlock.

Semáforos	b)
SA	3 2 1 2
SB	0 1 0
SC	2 1 0 1 0

Task	T1	T2	T3	T4	T5	T6	T7	T8	...T12	T13	T14	T15	T16	T17
TA	P(SA)	P(SA)	P(SA)	-	-	-	V(SB)	END						
TB	X	X	X	X	X	X	X	P(SB)	-	V(SC)	V(SA)	-	END	
TC	P(SC)	P(SC)	X	X	X	X	X	X	X	X	P(SC)	-	-	V(SB)

C) Deadlock pois tarefa C não executa. TA finaliza em T3, TB finaliza em T9.

Semáforos	c)
SA	2 1 0 1 0
SB	1 0 1
SC	1 0 1 0

Task	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T3
TA	P(SA)	P(SA)	X	X	X	X	X	P(SA)	-	-	-	V(SB)	END
TB	P(SB)	-	-	-	-	V(SC)	V(SA)	-	END				
TC	P(SC)	X	X	X	X	X	P(SC)	X	X	X	X	X	

D) Exite Deadlock, a tarefa TA não executa e fica bloqueada em T15. TB finaliza em T15 e TC finaliza em T9

Semáforos	d)
SA	0 1 0 1
SB	0 1 0 1
SC	3 2 1 0 1

Task	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15
TA	X	X	X	X	X	X	X	X	P(SA)	X	X	X	X	P(SA)	X
TB	X	X	X	X	X	X	P(SB)	-	-	-	-	V(SC)	V(SA)	-	END
TC	P(SC)	P(SC)	P(SC)	-	-	V(SB)	V(SB)	V(SA)	END						

E) Não tem deadlock, todas as task são executadas sem problemas. TA finaliza em T8. TB e TC finalizam em T9.

Semáforos	e)
SA	3 2 1 0
SB	1 0 1 2
SC	3 2 1 0 1

Task	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
TA	P(SA)	P(SA)	P(SA)	-	-	-	V(SB)	END						
TB	P(SB)	-	-	-	-	V(SC)	V(SA)	-	END					
TC	P(SC)	P(SC)	P(SC)	-	-	V(SB)	V(SB)	V(SA)	END					

F) Não tem deadlock, todas as task são executadas sem problemas. TA finaliza em T20. TB finaliza em T15. TC finaliza em T9

Semáforos	f)
SA	1 0 1 0 1
SB	0 1 0 1
SC	3 2 1 0 1

Task	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16
TA	P(SA)	X	X	X	X	X	X	X	P(SA)	X	X	X	X	P(SA)	-	-
TB	X	X	X	X	X	X	P(SB)	-	-	-	-	V(SC)	V(SA)	-	END	
TC	P(SC)	P(SC)	P(SC)	-	-	V(SB)	V(SB)	V(SA)	END							

T17	T18	T19	T20
-	-	V(SB)	END

G) Possui deadlock. Apenas de TB finaliza, em T9. TA fica bloqueado em T9 e TC em T8.

Semáforos	f)
SA	1 0 1
SB	1 0
SC	1 0 1

Task	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
TA	P(SA)	X	X	X	X	X	X	P(SA)	X					
TB	P(SB)	-	-	-	-	V(SC)	V(SA)	-	END					
TC	P(SC)	X	X	X	X	X	P(SC)	X						

2) A seguir, apresentamos uma nova sequência de operações do semáforo no início e no final das tarefas A, B, C. Considere que cada tarefa executa em um núcleo de processador dedicado. E considere que cada ação (P(Sx), V(Sx) ou .) possui tempo igual a 1T.

	Task A	Task B	Task C
1	P(SA)	P(SB)	P(SC)
2	P(SA)	P(SA)	P(SC)
3	V(SA)	.	P(SB)
4	.	.	.
5	.	.	.
6	.	P(SC)	V(SB)
7	V(SC)	V(SA)	V(SB)
8	END	END	V(SA)
9			END

Determine para os 6 casos a,b,c,d,e,f apresentados na tabela abaixo, se e em qual sequência as tarefas são executadas, usando as inicializações das variáveis do semáforo dadas na tabela.

Semáforos	a)	b)	c)	d)	e)	f)	g)
SA	2	1	2	0	3	2	1
SB	0	0	1	0	1	2	1
SC	2	1	1	2	1	2	1

A) Há deadlock, apenas a tarefa A executa em T8. TB fica bloqueado em T1 e TC em T3

Semáforos	A)
SA	2 1 0 1
SB	0
SC	2 1 0 1

Task	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13
TA	P(SA)	P(SA)	V(SA)	-	-	-	V(SC)	END					
TB	X	X	X	X	X	X	x	x					
TC	P(SC)	P(SC)	X	X	X	X	x	x					

2. Códigos em Python.

Nos seguintes códigos explique o comportamento do código e o conteúdo que será exibido ao final de sua execução.

```
#A
from threading import *
import time
l=Lock()
def wish(name,age):
    for i in range(3):
        l.acquire()
        print("Hi",name)
        time.sleep(2)
        print("Your age is",age)
        l.release()
t1=Thread(target=wish, args=("Sireesh",15))
t2=Thread(target=wish, args=("Nitya",20))
t1.start()
t2.start()
```

SAÍDA:

```
Hi Sireesh
Your age is 15
Hi Sireesh
Your age is 15
Hi Sireesh
Your age is 15
Hi Nitya
Your age is 20
Hi Nitya
Your age is 20
Hi Nitya
Your age is 20
```

- A) Neste exemplo é utilizado o Lock, ele irá bloquear o ‘recurso’ em uma thread e a próxima thread só irá ser executada quando a primeira thread que possui o recurso finalizar, dessa forma, liberando o recurso para a próxima thread. A intenção do Lock é justamente simular essa necessidade do recurso e um programa em espera.
-

```
#B
from threading import *
import time
s=Semaphore(2)
def wish(name,age):
    for i in range(3):
        s.acquire()
        print("Hi",name)
        time.sleep(2)
        s.release()
t1=Thread(target=wish, args=("Sireesh",15))
t2=Thread(target=wish, args=("Nitya",20))
t3=Thread(target=wish, args=("Shiva",16))
t4=Thread(target=wish, args=("Ajay",25))
t1.start()
t2.start()
t3.start()
t4.start()
```

SAÍDA:

```
Hi SireeshHi
  Nitya
HiHi  SireeshNitya
Hi Sireesh
Hi Nitya
Hi Shiva
Hi Ajay
Hi Shiva
Hi Ajay
Hi Shiva
Hi Ajay
```

- B) O código traz o uso de semáforos, as 4 threads são inicializadas com os parâmetros, porém as tarefas possuem apenas 2 contadores para permitir que apenas 2 threads sejam executadas simultaneamente.

```
#C
from threading import Lock, Thread
lock = Lock()
g = 0

def add_one():
    global g
    lock.acquire()
    g += 1
    lock.release()

def add_two():
    global g
    lock.acquire()
    g += 2
    lock.release()

threads = []
for func in [add_one, add_two, add_two, add_one,
add_one, add_two]:
    threads.append(Thread(target=func))
    threads[-1].start()

for thread in threads:
    thread.join()

print(g)
```

- C) O código acima modifica a mesma variável, a utilização do Lock é necessária para não haver confusões na execução pois sem o Lock, as threads iriam modificar a mesma variável podendo ter inconsistências no valor final.

3. Resolvendo problemas com Sincronização

A. A seguir é apresentado trecho de código Python. Analise o código e responda as seguintes questões:

- I. Explique a finalidade do código apresentado?
- II. Qual o resultado após execução do código?
- III. Execute o código 10 vezes. Os resultados foram iguais? Caso negativo, por qual motivo?
- IV. Utilize mecanismos de sincronização de forma que ao final da execução do código conta2 possua saldo 100 e conta1 possua saldo 0.

```
import threading
import time

class ContaBancaria():
    def __init__(self, nome, saldo):
        self.nome = nome
        self.saldo = saldo
    def __str__(self):
        return self.nome

conta1 = ContaBancaria("conta1", 100)
conta2 = ContaBancaria("conta2", 0)
class ThreadTransferenciaEntreContas(threading.Thread):
    def __init__(self, origem, destino, valor):
        threading.Thread.__init__(self)
        self.origem = origem
        self.destino = destino
        self.valor = valor

    def run(self):
        origem_saldo_inicial = self.origem.saldo
        origem_saldo_inicial -= self.valor
        time.sleep(0.001)
        self.origem.saldo = origem_saldo_inicial
        destino_saldo_inicial = self.destino.saldo
        destino_saldo_inicial += self.valor
        time.sleep(0.001)
        self.destino.saldo = destino_saldo_inicial

if __name__ == "__main__":

    threads = []
    for i in range(100):
        threads.append(ThreadTransferenciaEntreContas(conta1, conta2, 1))
    for thread in threads:
        thread.start()
    for thread in threads:
        thread.join()
    print('Saldo da', conta1, ':', conta1.saldo)
    print('Saldo da', conta2, ':', conta2.saldo)
```

- I. O código simula uma transferência entre duas contas bancárias por meio da utilização de threads, elas executam e modificam os saldos das contas.
- II. A execução mostra que foi passado algum valor da conta 1 para a 2, mas o valor passado não está bem definido, visto que as threads executam em

paralelo e modificam os mesmos valores, causando inconsistências nos valores.

- III. Sempre os resultados são diferentes, isso se dá pela execução paralela das thread e pela utilização do mesmo valor em processos diferentes o que normalmente pode trazer inconsistências no valor final.
- IV. Correção: Uso do Lock no método run() resolve e faz com o que a conta 1 zere e a conta dois fique com 100.

```
from threading import Thread, Lock
import time

class ContaBancaria():
    def __init__(self, nome, saldo):
        self.nome = nome
        self.saldo = saldo
    def __str__(self):
        return self.nome

conta1 = ContaBancaria("conta1", 100)
conta2 = ContaBancaria("conta2", 0)
l=Lock()

class ThreadTransferenciaEntreContas(Thread):
    def __init__(self, origem, destino, valor):
        Thread.__init__(self)
        self.origem = origem
        self.destino = destino
        self.valor = valor

    def run(self):
        l.acquire()
        origem_saldo_inicial = self.origem.saldo
        origem_saldo_inicial -= self.valor
        time.sleep(0.001)
        self.origem.saldo = origem_saldo_inicial
        destino_saldo_inicial = self.destino.saldo
        destino_saldo_inicial += self.valor
        time.sleep(0.001)
        self.destino.saldo = destino_saldo_inicial
        l.release()

if __name__ == "__main__":
```

```
threads = []
for i in range(100):
    threads.append(ThreadTransferenciaEntreContas(conta1, conta2,
1))
for thread in threads:
    thread.start()
for thread in threads:
    thread.join()
print('Saldo da', conta1, ':', conta1.saldo)
print('Saldo da', conta2, ':', conta2.saldo)
```