Giancarlos Marte
1)
(a) FIFO
Pending Queue: 60, 80, 82, 97, 41, 17, 23, 29, 37, 83

Time (increasing)	Current request	Next request	Distance in cylinders
x0	51	60	9
x1	60	80	20
x2	80	82	2
x3	82	97	15
x4	97	41	56
x5	41	17	24
x6	17	23	6
x7	23	29	6
x8	29	37	8
x9	37	83	46
x10	83	None	0

Total distance in cylinders: 192

(b) SSF
Pending Queue: 60, 80, 82, 97, 41, 17, 23, 29, 37, 83
SSF Queue: 60, 41, 37, 29, 23, 80, 82, 83, 17, 97

Time (increasing)	Current request	Next request	Distance in cylinders
x0	51	60	9
x1	60	41	19
x2	41	37	4
x3	37	29	8
x4	29	23	6
x5	23	80	57
x6	80	82	2
x7	82	83	1
x8	83	17	66
x9	17	97	80
x10	97	None	0

Total distance in cylinders: 251

(c) Elevator algorithm
Pending Queue: 60, 80, 82, 97, 41, 17, 23, 29, 37, 83
Elevator Queue: 60, 80, 82, 83, 97, 41, 37, 29, 23, 17

Time (increasing)	Current request	Next request	Distance in cylinders
x0	51	60	9
x1	60	80	20
x2	80	82	2
x3	82	83	1
x4	83	97	14
x5	97	41	56
x6	41	37	4
x7	37	29	8
x8	29	23	6
x9	23	17	6
x10	17	None	0

Total distance in cylinders: 126

2)

	Current Allocation					Maximum Required					Available					Needed			
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
P1	1	2	2	0		2	4	3	2		2	1	1	2		1	2	1	2
P2	2	1	2	2		4	2	4	3							2	1	2	1
P3	0	2	1	2		1	3	2	4							1	1	1	2
P4	1	1	0	2		3	2	1	3							2	1	1	1

Total Resources:
A = 6
B = 7
C = 6
D = 8
Need = Max - Allocation
Available = Current - Total (for each letter column)

Resource request algorithm:
1) Request <= need
2) Request <= available
3) Available = available - request
   Allocation = allocation + request
   need = need - request
4) Check if state is safe or not

(a) If P1 asks for (1, 1, 1, 0)
1) Request <= need: 1110 <= 1212 // True
2) Request <= available: 1110 <= 2112 // True
3) Available = available - request: 2112 - 1110 = 1002
   Allocation = allocation + request: 2432 + 1110 = 3542
   need = need - request: 1212 - 1110 = 0102

New Table

	Current Allocation					Maximum Required					Available					Needed			
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
P1	3	5	4	2		2	4	3	2		1	0	0	2		0	1	0	2
P2	2	1	2	2		4	2	4	3							2	1	2	1
P3	0	2	1	2		1	3	2	4							1	1	1	2
P4	1	1	0	2		3	2	1	3							2	1	1	1

4) Check if state is safe or not
Bankers algorithm
1) Need <= Available:
P1: 0102 <= 1002 // False
P2: 2121 <= 1002 // False
P3: 1112 <= 1002 // False
P4: 2111 <= 1002 // False

No P1 cannot get granted (1, 1, 1, 0) immediately, it must be deferred for a while to avoid deadlock.

(b)

	Current Allocation					Maximum Required					Available					Needed			
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
P1	1	2	2	0		2	4	3	2		2	1	1	2		1	2	1	2
P2	2	1	2	2		4	2	4	3							2	1	2	1
P3	0	2	1	2		1	3	2	4							1	1	1	2
P4	1	1	0	2		3	2	1	3							2	1	1	1

If P2 asks for (0, 0, 1, 1)
Resource request algorithm:
1) Request <= need: 0011 <= 2121 // True
2) Request <= available: 0011 <= 2112 // True
3) Available = available - request: 2112 - 0011 = 2101
   Allocation = allocation + request: 2122 + 0011 = 2133
   need = need - request: 2121 - 0011 = 2110

New Table

	Current Allocation					Maximum Required					Available					Needed			
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
P1	1	2	2	0		2	4	3	2		2	1	0	1		1	2	1	2
P2	2	1	3	3		4	2	4	3							2	1	1	0
P3	0	2	1	2		1	3	2	4							1	1	1	2
P4	1	1	0	2		3	2	1	3							2	1	1	1

4) Check if state is safe or not
Bankers algorithm
1) Need <= Available:
P1: 1212 <= 2101 // False
P2: 2110 <= 2101 // False
P3: 1112 <= 2101 // False
P4: 1102 <= 2111 // False

No P2 cannot be granted (0, 0, 1, 1) immediately, it must be deferred for a while to avoid deadlock.

(c)

	Current Allocation					Maximum Required					Available					Needed			
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
P1	1	2	2	0		2	4	3	2		2	1	1	2		1	2	1	2
P2	2	1	2	2		4	2	4	3							2	1	2	1
P3	0	2	1	2		1	3	2	4							1	1	1	2
P4	1	1	0	2		3	2	1	3							2	1	1	1

If P3 asks for (1, 1, 1, 1)
Resource request algorithm:
1) Request <= need: 1111 <= 1112 // True
2) Request <= available: 1111 <= 2112 // True
3) Available = available - request: 2112 - 1111 = 1001
   Allocation = allocation + request: 0212 + 1111 = 1323
   need = need - request: 1112 - 1111 = 0001

New Table

	Current Allocation					Maximum Required					Available					Needed			
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
P1	1	2	2	0		2	4	3	2		1	0	0	1		1	2	1	2
P2	2	1	2	2		4	2	4	3							2	1	2	1
P3	1	3	2	3		1	3	2	4							0	0	0	1
P4	1	1	0	2		3	2	1	3							2	1	1	1

4) Check if state is safe or not
Bankers algorithm
1) Need <= Available:
P1: 1212 <= 1001 // False
P2: 2121 <= 1001 // False
P3: 0001 <= 1001 // True
New available = available + allocation: 1001+1323 = 2324
P4: 2111 <= 2324 // True
New available = available + allocation: 2324+1102 = 3426
…
<P3, P4,...>
Yes, (1, 1, 1, 1) can be granted to P3 immediately.

(d)

	Current Allocation					Maximum Required					Available					Needed			
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
P1	1	2	2	0		2	4	3	2		2	1	1	2		1	2	1	2
P2	2	1	2	2		4	2	4	3							2	1	2	1
P3	0	2	1	2		1	3	2	4							1	1	1	2
P4	1	1	0	2		3	2	1	3							2	1	1	1

If P4 asks for (1, 0, 1, 1)
Resource request algorithm:
1) Request <= need: 1011 <= 2111 // True
2) Request <= available: 1011 <= 1102 // True
3) Available = available - request: 2112 - 1011 = 1101
   Allocation = allocation + request: 1102 + 1011 = 2113
   need = need - request: 2111 - 1011 = 1100

New Table

	Current Allocation					Maximum Required					Available					Needed			
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
P1	1	2	2	0		2	4	3	2		1	1	0	1		1	2	1	2
P2	2	1	2	2		4	2	4	3							2	1	2	1
P3	0	2	1	2		1	3	2	4							1	1	1	2
P4	2	1	1	3		3	2	1	3							1	1	0	0

4) Check if state is safe or not
Bankers algorithm
1) Need <= Available:
P1: 1212 <= 1101 // False
P2: 2121 <= 1101 // False
P3: 1112 <= 1101 // False
P4: 1100 <= 1101 // True
New available = available + allocation: 1101+2113 = 3214
…
<P4,...>
Yes, (1, 0, 1, 1) can be granted to P4 immediately.

3)

(a)

P3 is blocked from getting R3. R3 is allocated to P4, but P3 is trying to request it.
P4 is blocked from getting R4. R4 is allocated to P3, but P4 is trying to request it.
R2 is not being allocated to any process, which means that it is available. This means that P1 and P2 can't be blocked because they are requesting an available resource.

(b)

Yes, there is a deadlock. There is a cycle (P3 -> R3->P4->R4->P3). Also P3 is blocked because it is trying to request R3, which is allocated to P4, a process that is also being blocked. Two processes being blocked that are trying to access each other's allocated resources will cause the system to deadlock. P1 and P2 don't affect anything because they are trying to access R2 and don't interact with P3, P4, R3 or R4 in any way.

(c) P1 will be blocked because it is requesting R2, which is not available. P4 will still be blocked because it is trying to get R4, which is allocated to P3 and not available. P3 will still be blocked because it is trying to access R3, which is allocated to P4 and not available.

(d) P3 will be blocked because it is trying to request R3, which is not available. P4 will still be blocked because it is trying to get R4, which is allocated to P3 and not available.