1(a)

L^+ = L, N, O, M, Q, R, S, P

M^+ = M, P

N^+ = N, M, Q, R, P

O^+ = O, S

L is superkey, M, N, O are not superkey.

So, M → P, N → MQR, O → S violate BCNF.

1(b)

Split R by N → MQR, N^+ = MNPQR to

R1 = MNPQR

R2 = LNOS

Project FDs to R1

M	N	P	Q	R	closure	FD
√					M^+ = MP	M→P, not BCNF, split

Split R1 by M→P to

R3 = MP

R4 = MNQR

Project FDs to R3

M	P	closure	FD
√		M^+ = MP	M→P, M is superkey
	√	P^+ = P	nothing
√	√	MP^+ = MP	nothing

R3 is BCNF

Project FDs to R4

M	N	Q	R	closure	FD
√				M^+ = MP	nothing
	√			N^+ = MNPQR	N→MQR, N is superkey
		√		Q^+ = Q	nothing
			√	R^+ = R	nothing
√	√			irrelevant, because contains superkey	weaker
√		√		MQ^+ = MPQ	nothing
√			√	MR^+ = MPR	nothing
	√	√		irrelevant, because contains superkey	weaker
	√		√	irrelevant, because contains superkey	weaker
		√	√	QR^+ = QR	nothing
√	√	√		irrelevant, because contains superkey	weaker
√	√		√	irrelevant, because contains superkey	weaker
√		√	√	MQR^+ = MPQR	nothing
	√	√	√	irrelevant, because contains superkey	weaker
√	√	√	√	irrelevant, because contains superkey	nothing

R4 is BCNF

Project FDs to R2

L	N	O	S	closure	FD
√				L^+ = LMNOPQRS	L→NOS, L is superkey
	√			N^+ = MNPQR	nothing
		√		O^+ = OS	O→S, not BCNF, split

Split R2 by O→S to

R5 = OS

R6 = LNO

Project FDs to R5

O	S	closure	FD
√		O^+ = OS	O→S, O is superkey
	√	S^+ = S	nothing
√	√	OS^+ = OS	nothing

R5 is BCNF

Project FDs to R6

L	N	O	closure	FD
√			L^+ = LMNOPQRS	L→NO, L is superkey
	√		N^+ = MNPQR	nothing
		√	O^+ = OS	nothing
√	√		Irrelevant, because contains superkey	weaker
	√	√	NO^+ = NMOPQRS	nothing
√		√	irrelevant, because contains superkey	weaker
√	√	√	irrelevant, because contains superkey	nothing

R6 is BCNF

Final decomposition:

R3 = MP with FD M→P

R4 = MNQR with FD N→MQR

R5 = OS with FD O→S

R6 = LNO with FD L→NO

1(c)

New schema preserves dependencies.

Original dependencies: { L→NO, M→P, N→MQR, O→S}

New dependencies: { M→P, N→MQR, O→S, L→NO}

They are exactly same, so I don't need to do closure check to say new schema preserves dependencies.

1(d)

Assume (l,m,n,o,p,q,r,s) in R3 ⋈ R4 ⋈ R5 ⋈ R6, need to prove it must be in R.

By composition we know R has these elements:

L	M	N	O	P	Q	R	S
?	m	?	?	p	?	?	?
?	m	n	?	?	q	r	?
?	?	?	o	?	?	?	s
l	?	n	o	?	?	?	?

By M→P, we know if 2 rows both have m, they must both have p:

L	M	N	O	P	Q	R	S
?	m	?	?	p	?	?	?
?	m	n	?	p	q	r	?
?	?	?	o	?	?	?	s
l	?	n	o	?	?	?	?

By N→MQR, we know if 2 rows both have n, they must both have mqr:

L	M	N	O	P	Q	R	S
?	m	?	?	p	?	?	?
?	m	n	?	p	q	r	?
?	?	?	o	?	?	?	s
l	m	n	o	?	q	r	?

By O→S, we know if 2 rows both have o, they must both have s:

L	M	N	O	P	Q	R	S
?	m	?	?	p	?	?	?
?	m	n	?	p	q	r	?
?	?	?	o	?	?	?	s
l	m	n	o	?	q	r	s

By M→P again, we know if 2 rows both have m, they must both have p:

L	M	N	O	P	Q	R	S
?	m	?	?	p	?	?	?
?	m	n	?	p	q	r	?
?	?	?	o	?	?	?	s
l	m	n	o	p	q	r	s

In the last row of the table, we have (l,m,n,o,p,q,r,s). So any row in R3 ⋈ R4 ⋈ R5 ⋈ R6 is also in R, new schema is lossless.

2(a)

1. Split RHS get:

   ACD→E
   B→C
   B→D
   BE→A
   BE→C
   BE→F
   D→A
   D→B
   E→A
   E→C

2. Check if LHS has unnecessary attribute

   ACD→E:

   D^+ = A B C D E F so LHS reduce to D,
   Get D→E

   BE→A:

   B^+ = A B C D E F so LHS reduce to B,
   Get B→A

   BE→C:

   B^+ = A B C D E F so LHS reduce to B,
   Get B→C

   BE→F:

   B^+ = A B C D E F so LHS reduce to B,
   Get B→F

   Now we have:
   D→E
   B→C
   B→D
   B→A
   B→C
   B→F
   D→A
   D→B
   E→A
   E→C

3. Check for redundant FD

   D→E:

   Not redundant, because E not in other RHS
   Keep

B→C:

Redundant, because there is another B→C below.

Remove

B→D:

Not redundant, because B^+ = A B C F, without itself.

Keep

B→A:

Redundant, because B^+ = A B C D E F, without itself.

Remove

B→C:

Redundant, because B^+ = A B C D E F, without itself.

Remove

B→F:

Not redundant, because F not in other RHS

Keep

D→A:

Redundant, because D^+ = A B C D E F, without itself.

Remove

D→B:

Not redundant, because B not in other RHS

Keep

E→A

Not redundant, because E^+ = C E, without itself.

Keep

E→C

Not redundant, because E^+ = A E, without itself.

Keep

Finally we have minimal basis:

B→D
B→F
D→B
D→E
E→A
E→C

2(b)

	On LHS	On RHS	
G H	x	x	In every key
-	√	x	In every key
A C F	x	√	In no key
B D E	√	√	Need to check

B	D	E	Key	Closure
√			B G H	A B C D E F G H, key
	√		D G H	A B C D E F G H, key
		√	E G H	A C E G H, not key

No need to try other combination, because other combination will include at least one of B or D, so they are not key.

Keys are:

B G H

D G H

2(c)

1. Merge RHS, get:
   B→DF
   D→BE
   E→AC

2. Set relations
   R1(B, D, F), R2(D, B, E), R3(E, A, C)

3. No relation is superkey
   So, add R4(B, G, H)

Final set of relations:

R1(B, D, F), R2(D, B, E), R3(E, A, C), R4(B, G, H)

2(d)

It does not allow redundancy.
Because every FD is preserved in one and only one relation.
In every relations that have FD within(R4 doesn't have FD), the LHS of all FDs is the superkey of that relation.
Showing here:
In R1, LHS is B, B^+= BDF
In R2, LHS is D, D^+= BDE
In R3, LHS is E, E^+= ACE
In R4, there is no FD
So, it does not allow redundancy.