Problem 1

(a) a → b: **No**, b can be 2, 3, 4 when a is one.
(b) a → c: **Yes**, for every two a, c has the same value.
(c) b → a: **Yes**, all records have b in different value.
(d) b → c: **Yes**, the same reason.
(e) c → a: **No**, a can be 1 or 2 for the same value of c.
(f) c → b: **No**, b can be 2, 3, 4, 5 for the same value of c.
(g) ab → c: **Yes**. Each combination of ab is different, b can be different too.
(h) ac → b: **No**. For a=1 and c=2, b can be 2, 3 or 4.
(i) bc → a: **Yes**. Each combination of bc is different, a can be different too.

Problem 2

ac, bc, dc and ec are the candidate keys. Because each of them has an attribute closure that includes all the fields in R:

For ac:

$$a \to b \to d$$
$$d, c \to e$$

For bc:

$$b \to d$$
$$d, c \to e \to a$$

For dc:

$$d, c \to e \to a \to b$$

For ec:

$$e \to a \to b \to d$$

And removing any attribute from each key will not get the same complete closure.

Problem 3

(a) **af** is the candidate key, this can be illustrated by calculating its attribute closure:

$$a \to bcd$$
$$ad \to e$$
$$f \to gh$$

We can see $\{af\} \cup \{bcd\} \cup \{e\} \cup \{gh\} = \{abcdefgh\}$. And removing any attribute from $\{af\}$ will not have a compete closure set.

(b) **efg → h** can be removed. As we can see from above, FD: efg → h is not used to calculate the attribute closure of $\{af\}$. And we can still get the complete set of fields of R.

Problem 4

(a) **abe** is the candidate key. Its attribute closure:

$$a \to c$$
$$b \to d$$
$$de \to f$$

So $\{abe\} \cup \{c\} \cup \{d\} \cup \{f\} = \{abcdef\}$ and we cannot get the same attribute closure after we remove any attribute from this super key.

(b) We need to add e.g. **a → e** as the new FD to let *a* be a candidate key.
Now we have:

$$\begin{gathered} a \rightarrow bec \\ b \rightarrow d \\ de \rightarrow f \end{gathered}$$

So $\{a\} \cup \{bec\} \cup \{d\} \cup \{f\} = \{abcdef\}$ and we cannot get the same attribute closure after we remove any attribute from this super key.

Problem 5

(a) all nontrivial FDs:

- **a → b**
- **c → b**
- **ac → bc**
- **ac → b**
- **ac → c**

(b) First identify all the candidate keys, here we have *ac* as the only candidate key. Therefore, this instance of R is **not** in BCNF because of both **a → b** and **c → b** violate BCNF. (neither *a* nor *c* is not super key)

Use FD: a → b that violates BCNF of $R(\underline{a, c}, b)$, we now have:

$$\begin{gathered} R_0(\underline{a, c}) \\ R_1(\underline{a}, b) \end{gathered}$$

Which is in BCNF. Therefore we have it as one valid decomposition.

Problem 6

(a) Candidate keys: **ba, bc** and **bde**.

(b) For $R(a, b, c, d, e)$, we have two FDs from above that violate BCNF:

- **cd → e**
- **de → a**

Because left side of them are not super key.

(c) Use FD: cd → e, we now have:

$$\begin{gathered} R_0(\underline{a, b}, c, d) \\ R_1(\underline{c, d}, e) \end{gathered}$$

Use FD: bc → d that violates BCNF in R_0, we have R_0 decomposed:

$$\begin{gathered} R_2(\underline{a, b}, c) \\ R_3(\underline{b, c}, d) \end{gathered}$$

So **one possible decomposition** would be:

$$\begin{gathered} R_1(\underline{c, d}, e) \\ R_2(\underline{a, b}, c) \\ R_3(\underline{b, c}, d) \end{gathered}$$

On the other hand, use FD: de → a, R is decomposed into:

$$R_0(\underline{b, c}, d, e)$$
$$R_1(\underline{d, e}, a)$$

Use FD: dc → e that violates BCNF in R_0, decompose R_0 into:

$$R_2(\underline{b, c}, d)$$
$$R_3(\underline{d, c}, e)$$

Thus **another possible decomposition** is:

$$R_1(\underline{d, e}, a)$$
$$R_2(\underline{b, c}, d)$$
$$R_3(\underline{d, c}, e)$$

Problem 7

(a) The nontrivial FDs that capture the assumptions above:

- **s_ID, course_id, sec_id, semester, year → grade**
- **course_id, sec_id, semester, year → i_ID**
- **course_id, sec_id, semester, year → building, room_no**
- **course_id, sec_id, semester, year → time_slot_id**

(b) All candidate keys: **s_ID, course_id, sec_id, semester, year** (the only one).

(c) **No**, one instance of FD that violates BCNF is:

course_id, sec_id, semester, year → i_ID

Where course_id, sec_id, semester, year is not the super key in *takesSecTaughtBy*.

Given: course_id, sec_id, semester, year → i_ID

takesSecTaughtBy can be decomposed into:

- R0(s_ID, course_id, sec_id, semester, year, building, room_no, time_slot_id, grade)
- R1(course_id, sec_id, semester, i_ID)

Given: course_id, sec_id, semester, year → building, room_no

R0 can be decomposed into:

- R2(s_ID, course_id, sec_id, semester, year, time_slot_id, grade)
- R3(course_id, sec_id, semester, building, room_no)

Again, given: course_id, sec_id, semester, year → time_slot_id

R2 can be decomposed into:

- R4(s_ID, course_id, sec_id, semester, year, grade)
- R5(course_id, sec_id, semester, time_slot_id)

Thus one possible decomposition is:

- **R1(<u>course_id, sec_id, semester</u>, i_ID)**
- **R3(<u>course_id, sec_id, semester</u>, building, room_no)**
- **R4(<u>s_ID, course_id, sec_id, semester, year</u>, grade)**
- **R5(<u>course_id, sec_id, semester</u>, time_slot_id)**

(d) Relations not in BCNF may suffer from data redundancy, which can also be reflected by:

Update anomalies:
For example, if you want to change instructor for a given course section, you will need to update every records that has that section, because of redundancy.

Deletion anomalies:
If we want delete all the student information from a given course section, we will lose all the information associated with that course too, including building, room_no and time_slot_id etc.

Insertion anomalies:
When a course section needed to be inserted but no student has enrolled yet. In this case, there is no way to insert a record without knowing student ID.

(e) Based on the requirement, the appended FDs are:

- s_ID, course_id, sec_id, semester, year → grade
- course_id, sec_id, semester, year → i_ID (*)
- course_id, sec_id, semester, year → building, room_no
- course_id, sec_id, semester, year → time_slot_id
- course_id, s_ID → i_ID (*)
- i_ID, s_ID → course_id
- course_id → i_ID

Simplify the FDs above (by deleting those marked by star), we get:

- **s_ID, course_id, sec_id, semester, year → grade**
- **course_id, sec_id, semester, year → building, room_no**
- **course_id, sec_id, semester, year → time_slot_id**
- **i_ID, s_ID → course_id**
- **course_id → i_ID**

(f) Two candidate keys: **{course_id, sec_id, semester, year, s_ID}** and **{i_ID, sec_id, semester, year, s_ID}**.

(g) Use FD: course_id → i_ID, since course_id is not super key in:
takesSecTaughtBy(i_ID, s_ID, course_id, sec_id, semester, year, building, room_no, time_slot_id, grade)

We know *takesSecTaughtBy* is **not in BCNF**.

Now decompose it, use course_id → i_ID:

- R0(<u>s_ID, course_id, sec_id, semester, year</u>, building, room_no, time_slot_id, grade)
- R1(<u>course_id</u>, i_ID)

Similarly, we decompose R0 into:

- R2(<u>course_id, sec_id, semester, year</u>, time_slot_id)
- R3(<u>course_id, sec_id, semester, year</u>, building, room_no)
- R4(<u>s_ID, course_id, sec_id, semester, year</u>, grade)

Yields the decomposition in BCNF:

- **R1(<u>course_id</u>, i_ID)**
- **R2(<u>course_id, sec_id, semester, year</u>, time_slot_id)**
- **R3(<u>course_id, sec_id, semester, year</u>, building, room_no)**
- **R4(<u>s_ID, course_id, sec_id, semester, year</u>, grade)**

Problem 8

(a) It is **neither BCNF nor 3NF**. Because b is not super key and c is not part of super key (the only candidate key is *a* in this case).

One good 3NF decomposition would be:

- **R1(a, b)**
- **R2(b, c)**

It is in BCNF thus in 3NF, it is lossless because $R_1 \cap R_2 = \{b\}$ which is the primary key of R2. And it is dependency preserving because the two FDs are all satisfied apparently.

(b) First the candidate keys are: ab, cb and db.
Thus it is **not in BCNF** since either *c* or *d* is not super key. But it is **in 3NF** because both *c*, *d* and *a* are part of key.

(c) The only candidate key is *ab* in this case, so R is **neither in 3NF nor BCNF**. Because b is not a super key and c or d is not part of key.

One good 3NF decomposition would be:

- **R1(a, b)**
- **R2(b, c, d)**

It is in BCNF thus in 3NF, it is lossless because $R_1 \cap R_2 = \{b\}$ which is the primary key of R2. And it is dependency preserving because the two FDs are all satisfied apparently.

(d) Here candidate keys are: *ab*, *ad*, *bc* and *cd*. For all these four FDs, their right sides are all candidate keys, therefore the relation is in BCNF thus also in 3NF.