Given the following relation and functional dependencies:

```
R = (A, B, C, D, E, F)
FDs = { A -> C, A -> D, B -> E, B -> D, A -> A, AB -> A, A -> F }
```

1.1 Using 3NF, decompose R into the proper tables — Show your steps

(1)

A -> C

A -> D

B -> E

B -> D

A -> A

AB -> A

A -> F

(2)

A -> CDF

B -> ED

AB -> A

(3)

A -> CDF

B -> ED

ACDF // BED

1.2

(1)

A - > C

A - > D

B -> E

B -> D

A -> A

AB -> A

A -> F

C -> C

D -> D

E -> E

F -> F

(2)

A -> ACDF

B -> BDE

C -> C

D -> D

E -> E

F -> F

AB -> AB

(3)

A -> ACDF (non-trivial)

B -> BDE (non-trivial)

C -> C (trivial)

D -> D (trivial)

E -> E (trivial)

F -> F (trivial)

AB - > ABCDEF (super key)

Breaking on A

A + (ABCDEF – ACDF) = ABE

```
           ABCDEF
              |
      --------------------------
      |                        |
    ACDF                      ABE
```

Breaking on B

B + (ABE – BE) = BA

```
           ABCDEF
              |
      --------------------------
      |                        |
    ACDF                      ABE
                               |
                       -----------------
                       |               |
                       BE              BA
```

Thus, R(ABCDEF) becomes ACDF // BE // BA

```
R = (J, K, L, M, N)
FDs = { J -> JKLMN, K -> MN, N -> L, M -> N }
```

2.1 - Using 3NF, decompose R into the proper tables — Show your steps

(1)

J -> JKLMN

K -> MN

N -> L

M -> N

(2)

J -> KLMN

K -> MN

N -> L

M -> N

(3)

J -> K

K -> M

N -> L

M -> N

JK // KM // NL // MN

2.2 - Using BCNF, decompose R into the proper tables — Show your steps

(1)

J->JKLMN

K->MN

L->L

M->N

N->L

(2)

J->JKLMN

K->KMN

L->L

M->MN

N->NL

(3)

J->JKLMN (super key)

K->KMNL (non-trivial)

L->L (trivial)

M->MNL (non-trivial)

N->NL (non-trivial)

Breaking on K

K + (JKMNL – KMNL) = KMNL

```
             JKMNL
               |
       ------------------------
       |                      |
     KMNL                    JK
```

Breaking on M

M + (KMNL – MNL) = MNL

```
                JKMNL
                  |
          --------------------------
           |                      |
         KMNL                    JK
           |
   --------------------
    |                  |
   MNL               KM
```

Breaking on N

N + (MNL – NL) = MN

```
                JKMNL
                  |
          --------------------------
           |                      |
         KMNL                    JK
           |
   --------------------
    |                  |
   MNL               KM
     |
 -----------
  |        |
 LN      MN
```

Thus, R(JKLMN) becomes LN // MN // KM // JK

[40 points] Consider the relation `Courses(C, T, H, L, S, G)`, whose attributes may be thought of informally as `course_number`, `teacher_id`, `hour`, `location`, `student_id`, and `grade`. Let the set of FDs for Courses be

```
R = (C, T, H, L, S, G)
FDs = { C -> T, HL -> C, HT -> L, HS -> L, CS -> G }
```

Intuitively, we can interpret the FDs as

```
C -> T         a course has a unique teacher
HL -> C        only one course can meet in a given location at a given hour
HT -> L        a teacher can be in only one location at a given time (to teach a course)
HS -> L        a student can be in only one location at a given time (to attend a course)
CS -> G        a student gets only one grade in a course
```

3.1 - List all the superkeys for the Courses relation

Reasoning FDs (below), we can conclude that HS is the only superkey.

C -> T

HL -> CHLT

HT -> HTLC

HS -> HSLCTG

CS -> CSGT

3.2 - Verify that the given set of FDs is a minimal basis. Explain or discuss to show that it is (or is not) a minimal basis. (hint: Canonical Cover, Fc)

In order to verify that the given set of FDs is a minimal basis, we first check to see if any of the FDs can be removed. It appears we can remove HL -> C and still have a basis for the original set of FDs. Thus, it is not a minimal basis.

3.3

C -> T, HL -> C, HT -> L, HS -> L, CS -> G

First, we write all of the LHS: C -> , HL -> , HT -> , HS -> , CS -> .

Second, we copy the FDs as is: C -> T, HL -> C, HT -> L, HS -> L, CS -> G

Third, we check to see if there are any extraneous attributes on the left-hand side. For this step, we will look for any functional dependencies (FD) that have two or more attributes on the left-hand side and see if we can eliminate any. This involves the following examinations:

- The first FD we're going to look at is HL -> C. We will look at H closure (H+) and L closure (L+). H+ includes only H, and L+ includes only L. Since it appears that we cannot eliminate either the H or the L, there are no changes we can make to HL->C.
- The second FD we're going to look at is HT -> L. We will look at H closure (H+) and T closure (T+). H+ includes only H, and T+ includes only T. Since it appears that we cannot eliminate either the H or the T, there are no changes we can make to HT->L.
- The third FD we're going to look at is HS -> L. We will look at H closure (H+) and S closure (S+). H+ includes only H, and S+ includes only S. Since it appears that we cannot eliminate either the H or the S, there are no changes we can make to HT->S.
- The fourth FD we're going to look at is CS -> G. We will look at C closure (C+) and S closure (S+). C+ includes only C and T, and S+ includes only S. Since it appears that we cannot eliminate either the C or the S, there are no changes we can make to CS->G.

Since there are not extraneous attributes that can be eliminated, we are left with C -> T, HL -> C, HT -> L, HS -> L, CS -> G. This translates to CT // HLC // HTL // HSL // CSG.

3.4

To verify, we need to show that the decomposed relations satisfy:

<u>Lossless Join</u>

- It satisfies lossless join property, as you can take the intersection of C->T and HL->T, and find C which is a super key for C->T.

<u>Dependency Preservation</u>

- It can verify the dependency preserving properties, as C->T can be verified with CT, HL->C can be verified with HLC, HT->L can be verified with HTL, HS->L can be verified with HSL, and CS->G can be verified with CSG.

Since both properties are satisfied, it is in 3NF.

3.5

In order to verify, we need to show that the decomposed relations satisfy:

Lossless Join

In 3.4, we already verified that it satisfies the lossless join property.

For every non-trivial FD, X -> Attribute(s), X is a super key

- Consider CT, we know that C -> T. No non-key dependency in this decomposed relation.
- Consider HLC, we know that HL -> C. No non-key dependency in this decomposed relation.
- Consider HTL, we know that HT -> L. No non-key dependency in this decomposed relation.
- Consider HSL, we know that HS -> L. No non-key dependency in this decomposed relation.
- Consider CSG, we know that CS -> G. No non-key dependency in this decomposed relation.

Since both properties are satisfied, there is no BCNF violation.