Problem 1

R(A, B, C, D, E, F)

F = {
  FD1: A→ BC,
  FD2: C→ AD,
  FD3: DE → F
}

Part A

Decomposition(FD1)
  = FD4: A → B
Decomposition(FD2)
  = FD5: C → A
Transitivity(FD5, FD4)
  = FD6: C → B

Part B

Decomposition(FD1)
  = FD7: A → C
Decomposition(FD2)
  = FD8: C → D
Transitivity(FD7, FD8)
  = FD9: A → D
PseudoTransitivity(FD9, FD3)
  = FD10: AE → F

Problem 2

```
function ComputeAttrClosure(X, F)
   X⁺ := X;
   repeat unitl X⁺ has not changed
      foreach FD in F
         if FD Y → Z such that
               (i) Y is a subset of X⁺ , and
               (ii) Z is not a subset of X⁺ do
            X⁺ := X⁺ U Z
   return X⁺;
```

Part A

$X = \{A\}$
$X^+ = \text{ComputeAttrClosure}(X, F)$
$= \{A, B, C, D\}$

Part B

$X = \{C, E\}$
$X^+ = \text{ComputeAttrClosure}(X, F)$
$= \{C, E, A, D, F, B\}$
$= \{A, B, C, D, E, F\}$

Problem 3

R(A, B, C, D, E, F)

F = {
  FD1: AB → CDEF,
  FD2: E → F,
  FD3: D → B
}

Part A

import ComputeAttrClosure from “Problem 2”

Observations:

- A is only available on the left-hand side
  → Must be part of a candidate key (and a superkey)
- C and F are only available on the right-hand side
  → Cannot be part of a candidate key

Start with the largest set containing all attributes that must be part of a candidate key, as well as all attributes that can be part of a candidate key:

X = {A, B, D, E}
X^+ = ComputeAttrClosure(X, F)
  = {A, B, D, E, C, F}
  # superkey

Repeat for all permutations of superkeys with one of the non-mandatory attributes removed:
  → {A, B, D}, {A, B, E} , {A, D, E}

X = {A, B, D}
X_1^+ = ComputeAttrClosure(X, F)
  = {A, B, D, C, E, F}
  # superkey

X = {A, B, E}
X^+ = ComputeAttrClosure(Y, F)
  = {A, B, E, C, D, F}
  # superkey

X = {A, D, E}
X^+ = ComputeAttrClosure(Y, F)
  = {A, D, E, F, B, C}
  # superkey

Repeat for all permutations of superkeys with one of the non-mandatory attributes removed:
→ {A, B}, {A, D}, {A, E}

X = {A, B}
X^+ = ComputeAttrClosure(X, F)
= {A, B, C, D, E, F}
superkey

X = {A, D}
X^+ = ComputeAttrClosure(X, F)
= {A, D, B, C, E, F}
superkey

X = {A, E}
X^+ = ComputeAttrClosure(Y, F)
= {A, E, F}
not superkey

Repeat for all permutations of superkeys with one of the non-mandatory attributes removed:
→ {A}

X = {A}
X^+ = ComputeAttrClosure(X, F)
= {A}
not superkey

Resulting candidate keys:
→ {A, B}, {A, D}

Part B

FD2 and FD3, since the left-hand sides are not superkeys (a superkey must be a superset of a candidate key).

Part C

- Let X → Y be the FD that violates BCNF in a relation schema R
- Replace R by two new relation schemas R1 and R2 constructed as follows
- Create R1 with all the attributes in X and in Y
- Create R2 from R by removing all attributes that are in Y and not in X

R(A, B, C, D, E, F)

F = {
 FD1: AB → CDEF,
 FD2: E → F,
 FD3: D → B
}

Candidate keys:
 {A, B}, {A, D}

- Let R be a relation schema with a set F of FDs
- Let R1 , R2 , ..., Rn be a decomposition of R
- For every Ri we call the set of all FDs in F^+ that mention only attributes from Ri the restriction of F to R i
- Then, the decomposition is dependency preserving if for the restrictions F1 , F2 , ..., Fn of F to R1 , R2 , ..., Rn it holds that:
  $(F1 \cup F2 \cup ... \cup Fn)^+ = F^+$

First we decomposite R on the violating dependency FD2:

R1(E, F)
 FD2: E → F
 Candidate keys: {E}
 # BCNF

R2(A, B, C, D, E)
 Decomposition(FD1)
 = FD4: AB → CDE
 FD3: D → B
 Candidate keys: {A, B}, {A, D}
 # not BCNF

Then we decomposite R2 on the violating dependency FD3:

R2A(D, B)
 FD3: D → B
 Candidate keys: {D}
 # BCNF

R2B(A, C, D, E)
 PseudoTransitivity(FD3, FD1)
 = FD5: AD → CEF
 Candidate keys: {A, D}
 # BCNF

Lastly, we verify that the functional dependencies are preserved:

F1 U F2 U F3 = {
E → F
D → B
AD → CEF
} # where Fn is the functional dependencies of Rn

Since **B** is only on the right hand side, we can conclude that we cannot use **B** to deduce any other attributes. Hence, we do not have dependency preservation.

Problem 4

R(A, B, C, D, E)

F = {
FD1: ABC → DE,
FD2: BCD → AE,
FD3: C → D
}

Part A

import ComputeAttrClosure from "Problem 2"

Observations:

- B and C are only available on the left-hand side
  → Must be part of a candidate key (and a superkey)
- E is only available on the right-hand side
  → Cannot be part of a candidate key

Start with the largest set containing all attributes that must be part of a candidate key, as well as all attributes that can be part of a candidate key:

X = {B, C, A, D}
X^+ = ComputeAttrClosure(X, F)
= {B, C, A, D, E}
superkey

Repeat for all permutations of superkeys with one of the non-mandatory attributes removed:
→ {A, B, D}, {A, B, E} , {A, D, E}

X = {B, C, A}
X^+ = ComputeAttrClosure(X, F)
= {B, C, A, D, E}

\# superkey

X = {B, C, D}
X^+ = ComputeAttrClosure(X, F)
= {B, C, D, A, E}
\# superkey

Repeat for all permutations of superkeys with one of the non-mandatory attributes removed:
→ {A, B, D}, {A, B, E} , {A, D, E}

X = {B, C}
X^+ = ComputeAttrClosure(X, F)
= {B, C, D, A, E}
\# superkey

Resulting candidate keys:
→ {B, C}

FD3 violates BCNF, since the left-hand side is not a superkey (a superkey must be a superset of a candidate key).

Part B

- Let X → Y be the FD that violates BCNF in a relation schema R
- Replace R by two new relation schemas R1 and R2 constructed as follows
- Create R1 with all the attributes in X and in Y
- Create R2 from R by removing all attributes that are in Y and not in X

- Let R be a relation schema with a set F of FDs
- Let R1 , R2 , ..., Rn be a decomposition of R
- For every Ri we call the set of all FDs in F^+ that mention only attributes from Ri the restriction of F to R i
- Then, the decomposition is dependency preserving if for the restrictions F1 , F2 , ..., Fn of F to R1 , R2 , ..., Rn it holds that:
  $(F1 \cup F2 \cup ... \cup Fn)^+ = F^+$

R(A, B, C, D, E)

F = {
FD1: ABC → DE,
FD2: BCD → AE,
FD3: C → D
}

Candidate keys:
{B, C}

First we decomposite R on the violating dependency FD3:

R1(C, D)
FD3: C → D
Candidate keys: {C}
\# BCNF

R2(A, B, C, E)
  PsuedoTransitivity(FD3, FD2)
    = FD4: BC → AE
  Candidate keys: {B, C}
  # BCNF

Lastly, we verify that the functional dependencies are preserved:

F1 U F2 = {
  C → D
  BC → AE
} # where Fn is the functional dependencies of Rn

Trivially, <superset of BC> → AE, hence:
  BCD → AE

Trivially, BC → C
Transitivity(BC → C, C → D)
  = BC → D
Union(BC → D, BC → AE)
  = BC → ADE
Decomposition(BC → ADE)
  = BC → DE
Trivially, <superset of BC> → DE, hence:
  ABC → DE

Since we have {**C → D**, **BCD → AE** and **ABC → DE**}, the original FD's in R's F, it follows that the functional dependencies are preserved.