Problem 1 Consider a schema with attributes $X, Y, Z, U, V, W$ and FDs

$$Z \rightarrow W, V \rightarrow X, VZ \rightarrow U, X \rightarrow Y.$$ 

1. How many candidate keys does it have? Explain your answer.

Answer: One, $VZ$. $V, Z$ don’t appear in the rhs’s of FDs, hence any key must contain them. And closure of $VZ$ is $XYZUVW$.


Answer: You get a decomposition by using $V \rightarrow XY$ first, getting $VXY$ with $X \rightarrow Y$ as a non-key FD, and $V \rightarrow XY$. Apply $X \rightarrow Y$ and get the schemas $(XY, X \rightarrow Y), (XV, V \rightarrow X)$.

The other set (after using $V \rightarrow XY$) is $VZUW$, with $Z \rightarrow W$ being non-key FD. Decomposing we get $(ZW, Z \rightarrow W)$ and $(VZU, VZ \rightarrow U)$. All schemas are in BCNF, all FDs are preserved.

Problem 2 Consider a schema with attributes $A, B, C, D, E, F$ and FDs

$$A \rightarrow C, AB \rightarrow C, C \rightarrow DF, CD \rightarrow F, CE \rightarrow AB, EF \rightarrow C.$$ 

(a) Find a minimum cover for this set of FDs.

Answer: $A \rightarrow C, C \rightarrow D, C \rightarrow F, CE \rightarrow A, CE \rightarrow B, EF \rightarrow C$.

(b) Find a lossless dependency-preserving 3NF decomposition of the schema.

Answer: $(AC, A \rightarrow C), (CDF, C \rightarrow DF), (ABCE, CE \rightarrow AB), (CEF, EF \rightarrow C)$.

Problem 3 Consider a relational schema with attributes $A, B, C, D$ and functional dependencies $AB \rightarrow CD, D \rightarrow C, B \rightarrow C$.

Does this schema have a lossless dependency-preserving BCNF decomposition?

If yes, present such a decomposition; if no, explain why, and find a lossless dependency-preserving 3NF decomposition.

Answer: $AB$ is a key, the other FDs violate BCNF. Using either one of them for the decomposition algorithm, one gets a schema that loses one of those FDs.

To decompose into 3NF, note that $AB \rightarrow D, D \rightarrow C, B \rightarrow C$ is a minimum cover, and hence decomposition is $(ABD, AB \rightarrow D), (CD, D \rightarrow C), (BC, B \rightarrow C)$.

Problem 4 Is the following schedule conflict serializable? Explain why. If it is, give an equivalent serial schedule.

$$| T_1 | T_2 | T_3 | T_4 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>read(X)</td>
<td>write(X)</td>
<td>read(X)</td>
<td>write(Z)</td>
</tr>
<tr>
<td>read(Y)</td>
<td>write(Y)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>read(Z)</td>
<td>write(Z)</td>
<td>read(Z)</td>
<td>write(Z)</td>
</tr>
</tbody>
</table>
Answer: The precedence graph has edges $t_1 \rightarrow t_2$, $t_1 \rightarrow t_3$, $t_1 \rightarrow t_4$, $t_2 \rightarrow t_3$, $t_3 \rightarrow t_4$. There are no cycles, hence it is conflict serializable. Topological sort on this graph gives us $t_1, t_2, t_3, t_4$ which is an equivalent serial schedule.

**Problem 5** Suggest a change in the previous schedule that makes it *non-serializable.*

Answer: there are many, just introduce any cycle in the previous graph. For instance, make $\text{read}(Z)$ in $t_4$ earlier.

**Problem (only if there is time)** Let $(U, F)$ be a relational schema, where $U$ is a set of attributes and $F$ is a set of functional dependencies, and $K_1, \ldots, K_n$ be its candidate keys. We know that for any $1 \leq i < j \leq n$, $K_i \not\subseteq K_j$ and $K_j \not\subseteq K_i$, by the definition of candidate keys.

Now your goal is to check if the converse is true: given $K_1, \ldots, K_n \subseteq U$ satisfying the above condition, is there a relational schema $(U, F)$ whose candidate keys are precisely $K_1, \ldots, K_n$?

If the answer is no, give a counterexample; if the answer is yes, give a proof.

No counterexample or proof longer than 10 lines will be accepted.

Answer: Yes. $F$ contains $K_i \rightarrow U$ for all $i \leq n$. By the algorithm CLOSURE, if $X$ does not contain one of $K_i$s, then $C_F(X) = X$ and $X$ is not a key, if $K_i \subseteq X$, then $C_F(X) = U$. So the candidate keys are precisely the minimal sets among $K_1, \ldots, K_n$, and since no $K_i$ is contained in $K_j$, these are exactly $K_1, \ldots, K_n$. 

2