Answer 19.10

1. Candidate key(s): BD. The decomposition into BC and AD is unsatisfactory because it is lossy (the join of BC and AD is the cartesian product which could be much bigger than ABCD)

2. Candidate key(s): AB, BC. The decomposition into ACD and BC is lossless since ACD $\cap$ BC (which is C) $\rightarrow$ACD. The projection of the FD's on ACD include C $\rightarrow$D, C $\rightarrow$A (so C is a key for ACD) and the projection of FD on BC produces no nontrivial dependencies. In particular this is a BCNF decomposition (check that R is not!). However, it is not dependency preserving since the dependency AB $\rightarrow$C is not preserved. So to enforce preservation of this dependency (if we do not want to use a join) we need to add ABC which introduces redundancy. So implicitly there is some redundancy across relations (although none inside ACD and BC).

3. Candidate key(s): A, C. Since A and C are both candidate keys for R, it is already in BCNF. So from a normalization standpoint it makes no sense to decompose R further.

4. Candidate key(s): A. The projection of the dependencies on AB are: A $\rightarrow$B and those on ACD are: A $\rightarrow$C and C $\rightarrow$D (rest follow from these). The scheme ACD is not even in 3NF, since C is not a superkey, and D is not part of a key. This is a lossless-join decomposition (since A is a key), but not dependency preserving, since B $\rightarrow$C is not preserved.

5. Candidate key(s): A (just as before) This is a lossless BCNF decomposition (easy to check!) This is, however, not dependency preserving (B consider $\rightarrow$C). So it is not free of (implied) redundancy. This is not the best decomposition (the decomposition AB, BC, CD is better.)