Consider the table:
STAFF_MEETING (EmployeeName, ProjectName, Date)
The rows of this table record the fact that an employee from a particular project attended a meeting on a given date. Assume that a project meets at most once per day. Also, assume that only one employee represents a given project, but that employees can be assigned to multiple projects.

- **State the functional dependencies in STAFF_MEETING.**
- (ProjectName, Date) -> EmployeeName, ProjectName
- (ProjectName) -> EmployeeName

- **Transform this table into one or more tables in BCNF. State the primary keys,**
  candidate keys, foreign keys, and referential integrity constraints.
- Candidate Key- ProjectName, Date, EmployeeName
- PROJECT_EMPLOYEE(ProjectName, EmployeeName)
- a. Primary key-EmployeeName
- b. Foreign key- ProjectName
- PROJECT_DATE(ProjectName, Date)
- a. Primary key- ProjectName
- b. Constraint- ProjectName in PROJECT_EMPLOYEE must exist in PROJECT_DATE.

- **Is your design in part B an improvement over the original table? What advantages**
  and disadvantages does it have?
- Yes.
- Advantage- It simplifies and gets rid of redundant data.
- Disadvantage- Name can be entered after date and project is set.

Consider the table:
STUDENT (StudentNumber, StudentName, Dorm, RoomType, DormCost, Club, ClubCost, Sibling, Nickname)
Assume that students pay different dorm costs, depending on the type of room they have, but that all members of a club pay the same cost. Assume that students can have multiple nicknames.

- **State any multivalued dependencies in STUDENT.**
- StudentNumber ->-> StudentName
- StudentNumber ->-> Nickname
- StudentNumber ->-> Club

- **State the functional dependencies in STUDENT.**
- StudentNumber -> (StudentName, Dorm, RoomType, Sibling)
- RoomType -> DormCost

- Club -> ClubCost

- **Transform this table into two or more tables such that each table is in BCNF and in**
  4NF. State the primary keys, candidate keys, foreign keys, and referential integrity constraints.
- Candidate key- Number, RoomType, Club
- TABLE_11(Number, Name, Dorm, Sibling, RoomType)
- a. Primary key- Number
- b. Foreign key- RoomType
- c. Constraint- RoomType must be in TABLE_12
- TABLE_12(RoomType, DormCost)
- a. Primary key- RoomType
- TABLE_21(Number, Club)
- a. Primary key- Number
- b. Foreign key- Club
- c. Constraint- Number must be in TABLE_11, Club must be in TABLE_22
- TABLE_22(Club, ClubCost)
- a. Primary key- Club
- TABLE_3(Number, Nickname)
- a. Primary key- Number
- c. Constraint- Number must be in TABLE_11

- **Assume that Marcia keeps a table of data about her customers. Consider just the follow-**
  ing part of that table:
  CUSTOMER (Phone, FirstName, LastName)
  Explain the conditions under which each of the following are true:

  1. **Phone : (FirstName, LastName)**
     Phone number is unique

  2. **(Phone, FirstName) : LastName**
     Phone number is unique

  3. **(Phone, LastName) : FirstName**
     Phone number is unique

4. **(LastName, FirstName) : Phone**
   LastName+FirstName is unique

5. **Phone : : LastName**
   There are multiple last names for the same number

6. **Phone : : FirstName**
   There are multiple first names for the same number

7. **Phone : : (FirstName, LastName)**
   There are multiple first and last names for the same number

- **Is condition A.7 the same as conditions A.5 and A.6? Why or why not?**
  No, 7 relies on both FirstName and LastName having multiple values for the same number.

- **State an appropriate referential integrity constraint for the tables: CUSTOMER (Phone, FirstName, LastName) INVOICE (InvoiceNumber, DateIn, DateOut, Phone)**

  INVOICE.Phone must be in CUSTOMER

- **Consider the tables:**
  CUSTOMER (Phone, FirstName, LastName)
  INVOICE (InvoiceNumber, DateIn, DateOut, FirstName, LastName) What does the following referential integrity constraint mean?
  INVOICE.(FirstName, LastName) must be in CUSTOMER.(FirstName, LastName)

FirstName and LastName makes a composite key for INVOICE and both columns have to be in CUSTOMER

Is this constraint the same as the set of referential integrity constraints: INVOICE.FirstName must be in CUSTOMER.FirstName INVOICE.LastName must be in CUSTOMER.LastName
Explain why or why not.

Different because they no longer make a composite key.

- **Do you prefer the design in C or the design in D? Explain your reasoning.**

I like C more. It is much less complicated.

- **Transform the following table into two or more tables in BCNF and 4NF. Indicate the primary keys, candidate keys, foreign keys, and referential integrity constraints. Make and state assumptions as necessary.**
  INVOICE (CustomerNumber, FirstName, LastName, Phone, InvoiceNumber, DateIn, DateOut, ItemType, Quantity, ItemPrice, ExtendedPrice, SpecialInstructions)

Candidate keys- CustomerNumber, FirstName, Phone, InvoiceNumber, ItemType

CUSTOMER(CustomerNumber, FirstName, LastName, Phone, InvoiceNumber)

Primary key- CustomerNumber

Foreign key- InvoiceNumber

Constraint- InvoiceNumber has to be in INVOICE

INVOICE(InvoiceNumber, DateIn, DateOut, ItemType)

Primary key- InvoiceNumber

Foreign key- ItemType

Constraint- ItemTpye has to be in INVOICE_ITEM

INVOICE_ITEM(ItemType, Quantity, ItemPrice, ExtendedPrice, SpecialInstructions)

Primary key- ItemType

- **Explain how your answer to question F changes depending on whether you assume that CustomerNumber : (FirstName, LastName)**
  or
  CustomerNumber : : (FirstName, LastName)

If there is a multivalued dependency, another table would need to be created to fulfill 4NF and BCNF