Benji (Benjamin) Goldstein and Tyler Benson
1)
a) Abstraction is distilling real life implications in to implementable ideas.
b) Encapsulation is having each class know exactly enough to preform its one task and no more. Often it should move tasks to other classes to preform their one task. From the book: Typically defined as data hiding, but better thought of as any kind of hiding.
c) Cohesion is roughly applied encapsulation. Its how well each object preforms exactly one goal. How closely the operations in a routine are related.
d) Coupling is how interconnected objects are. The goal is low coupling since it means that encapsulation is functioning properly and able to preform its job.
e) Abstraction is key in all coding and object oriented is no exception. Good abstraction is only recording meaningful data, there is no reason for cu canvas to know about your drivers license but they should know your student id. Generally public variables are bad encapsulation, global variables especially. Good encapsulation keeps variables within scope either using protected or private as the situation demands. Cohesion is well applied when each class does exactly one thing. Functional decomposition is not cohesive since the main function does almost everything. A single main function is highly coupled, since it has lots of interactions, but that isn't the goal of OOD. Instead having a few, possibly one, way of messaging a class is better.

2)
Assumptions: Taxes, bonuses, and overtime exist. There is a table that contains active employees. There is a table that contains hours worked. There is a table that contains pay per hour. There is a table that contains amount paid per pay period. No one is salary based. We are legally able to automatically write checks, ie there is no direct deposit.

- Connect to Database
- Get List of employees
- For each employee find hours worked
    - Apply special payments for hours worked (overtime)
- Apply bonuses
- Apply taxes
- For each employee add total paid to database
- For each employee print paycheck

3)
Classes:

Controller Class: Knows payment period, employee lists, which version or defaults to use for taxes, special payment, and paycheck class, which database to connect to.

Paycheck Class: Knows how to print a pay check. Can ask employee object for required information. Can send information to be written to Database connection class, through database row class.

Special Payment Class [static]: Knows how to apply bonuses and overtime

Taxes Class [static]: Knows how to apply taxes to amount owed.

Employee class: Knows employee id. Knows hours rate, current owed value (including taxes). Can ask a database row object for additional information.

Database Row class: Knows how to consume a row from the database, subclasses for each table

Database Connection class: Knows how to provide a database row class with a single row from a database. Knows how to connect to the database.

Instantiation Order:

1. Instantiate Controller class
2. Create Database Connection, have it connect
3. Instantiate Taxes, Special Payment, and Paycheck classes
4. For each employee in database, create Employee class.
    a. Each Employee does the following
    b. Asks Database (with database row) to get payment information for payment period (asking Controller)
    c. Call special payment class; to add special payments
    d. Call taxes class; to add taxes
    e. Call paycheck class; to print