Page 2 (1, 2)

1. A. (6) Describe the roles of the MVC architectural level pattern

   Answer: (+2 each)

   Model - Complete, self-contained representation of object managed by the application, provides a number of services to manipulate the data, supports computation and persistence

   View - Responsible for a particular presentation of information from the model, may store values that are needed for a particular presentation of the data

   Controller - Responsible for interaction issues, may lead to changes in the model, may lead to changes in the view

   B. (3) How is the MVC architectural-level pattern used in Django (include how it is similar and dissimilar)?

   Answer: (+3)

   Model is generally the same

   Django's View is the Controller in the MVC framework.

   Django's Template serves the role of the View in the MVC framework

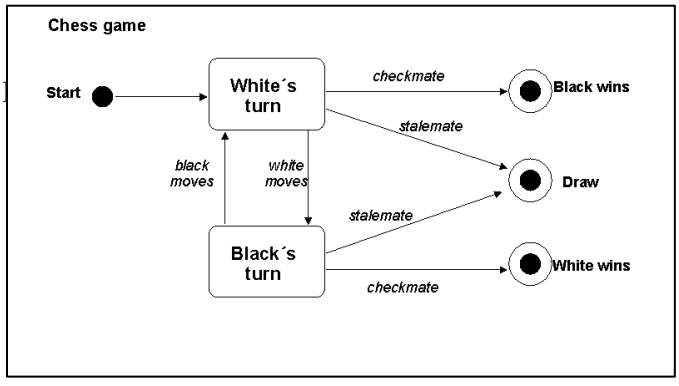
2. A. (3)What is the name of the UML diagram shown in Figure 1?

   Answer: (+3)
   State diagram or State Machine diagram

   B. (4) What does this type of diagram generally depict?

   Answer: (+4 for full points must include more than it depicts the state of an object)

   UML state machine diagrams depict the various states that an object may be in and the transitions between those states. {A state represents a stage in the behavior pattern of an object.}



Page 3 (3, 4)

3. A. (4) When using a distributed version control system like Git what must take place for others on your team to see your changes?

Answer (+1 for each or valid description)
You commit
You push
They pull
They update

B. (6) In addition to incorporating others' changes frequently, describe three other best practices when using a version control system.

Answer: (+2 for each best practice (common ones listed below however, could be others)

Use a descriptive commit message
Make each commit a logical unit
Don't commit generated files
Share your changes frequently
Coordinate with your co-workers

C. (3)What is the purpose of branching in Git?

Answer: (+3)
The purpose of branching in GIT is that you can create your own branch and jump between those branches. It will allow you to go to your previous work keeping your recent work intact

D. (2) True/**False** Git can handle all merge conflicts automatically.

4. A. (2) When referring to modules, what is coupling?
Answer: (+2)
Coupling is a measure that defines the lever of inter-dependability among modules of a program. It tells at what level the modules interfere and interact with each other.

B. (6) Why would a programmer strive for loosely coupled modules (give 2 reasons)?
Answer: (+3 for each reason)

Loose coupling makes it possible to:
Understand one class without reading others
Change one class without affecting others
Thus: improves maintainability

Page 4 (5)

5. A. (2) What is the name of UML diagram shown in Figure 2?

Answer: (+2)

Class

B. (2) Describe the association between Customer and Order (include the multiplicity in your description)

Answer: (+2)

One Customer can place one or more orders

C. (2) What does the "–" represent and why is it used?

Answer: (+2)

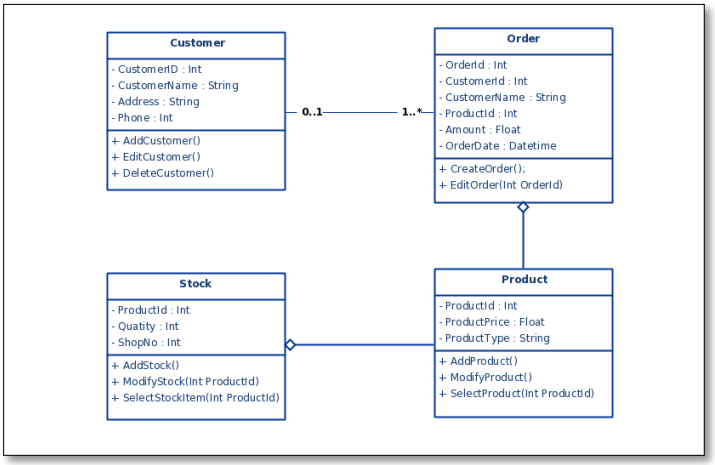
"-" represents private attributes are typically private so other classes cannot access them

D.(2) What does the "+" represent and why is it used?

Answer: (+2)

"+" represents public methods are normally public so, they can be accessible outside the class

Figure 2



Page 5 (6, 7)

6. (16) Matching

Match the term to the correct definition or example		
E	1. Information hiding	A. Describes a design structure that solves a particular design problem within a specific context.
H	2. Separation of Concerns	B. At the highest level, a solution is state in broad terms using the language of the problem environment. At the lower levels, a more detailed description of the solution is provided.
D	3. Stepwise Refinement	C. Software is divided into separately named and addressable components that are integrated to satisfy problem requirements.
G	4. Refactoring	D. Development by successfully filtering levels of procedural detail.
F	5. Architecture	E. Principle of segregation of the design decisions in a computer program that are most likely to change, thus protecting other parts of the program from extensive modification if the design decision is changed.
B	6. Abstraction	F. The structure or organization of program components the manner in which these components interact and the structure of data used by those components.
A	7. Patterns	G. Process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure
C	8. Modularity	H. Any complex problem can be more easily handled if it is subdivided into pieces that can each be solved and/or optimized independently

7. (9) Discuss the advantages and disadvantages of modular programming (include 3 advantages and 3 disadvantages in your discussion)

Answer: (+1.5 for each advantage, +1.5 for each disadvantage)

Advantages (may be others if it is a logical advantage)

1. Small problems are easier to understand
2. Easier to test small modules
3. Debugging is easier with small modules
4. Easier to update and maintain small modules
5. Team members can work on different modules
6. Code reusability – modules can be used in multiple programs

Disadvantages (may be others if it is a logical disadvantage not just negating an advantage)

Modules need to be linked together

Cross referencing the modules

Takes more time -time it takes to work out the modules, interfaces, test modules and links between modules and links

Page 6 (8, 9)

8. A. (2) When referring to modules, what is cohesion?

Answer: (+2)

Cohesion is a measure that defines the degree of intra-dependability within elements of a module.

B. (6) Why would a programmer strive for highly cohesive modules (give 2 reasons, not the same ones you gave in the question 4B)?

Answer: (+3 for each reason)

High cohesion makes it easier to:

Understand what a class or method does

Use descriptive names

Reuse classes or methods

9. A. (4) Discuss 2 roles a database management system (DBMS) can play in database development?

Answer: (+2 each for any 2 of the following)

Intermediary between the user and the database

Enables data to be shared

Presents the end user with an integrated view of the data

Receives and translates application requests into operations required to fulfill the requests

Hides database's internal complexity from the application programs and users

B. (6) Describe 3 ways a DBMS can improve data in a database (must be different than the roles given in the previous question).

Answer: (+ 2 each for any 3 of the following)

Improves:

Data sharing

Data security

Data access

Decision making

Data quality: Promoting accuracy, validity, and timeliness of data

Page 7 (10)

10. A manufacturing company produces products. Product information stored is product name, id, and quantity on hand. These products are made up of many components. Each component can be supplied by one or more suppliers. Component information kept is component id, name, description, suppliers who supply them, and which products they are used in.

Create an ERD (using the Chen or Crow's foot notation) to show how you would track this information. Show entity names, primary keys, attributes for each entity, relationships between the entities and cardinality.
Assumptions:

- A supplier can exist without providing components.
- A component does not have to be associated with a supplier.
- A component does not have to be associated with product.
- Not all components are used in products.
- A product cannot exist without components.

Answer: (+1 for each correct entity identified (5), +.25 for correct cardinality (can change slightly if assumptions are made however, no many to many relationships) (8), +..25 for primary keys identified for each entity, +.25 for each attribute)

5	entities +1 for each (5 entities)
2	cardinality +.25 for each (8 cardinality notations)
1.75	primary key +25 for each (7 primary keys)
1.25	attributes +.25 for each (Component -> CompName, Description, Supplier -> SupplierName, Product -> ProdName, QtyOnHand (5 attributes)