Problem 2

- The transportation class is an abstract class because different types of transportation have different properties such as speed, engine properties, etc. The types of transportation such as horse, airship, and ship classes are not abstract. Horse, airship, and ship classes inherit from the transportation class because they all are types of transportation. Each inherited class needs to implement its own option of speed and engine properties, therefore making transportation abstract makes most sense. The color option should also be in the transportation class as a data member and it should be public inheritance.

Transportation abstract class
Horse class ->(inherits from) transportation class
Ship class->(inherits from) transportation class
Airship class->(inherits from) transportation class

- The terrain class is abstract. Water, mountains, plains, and forest classes are not abstract. Those four classes inherit from terrain because they are all different types of terrain, and those classes have "is-a" relationship with terrain. Terrain is an abstract class because different types of terrain have different restrictions on the types of transportation allowed. As a result, each inherited class must implement its own unique possible restriction. This should be done through public inheritance.

Terrain abstract class
Water class->(inherits from) terrain class
Mountains class-> (inherits from) terrain class
Forest class-> (inherits from) terrain class
Plains class (inherits from) terrain class

- The Item class is abstract. Healing potion, magic potion, tent, and phoenix down classes are not abstract. Armor and weapons classes are abstract. Shield, helmet, chain mail, gauntlets, sword, spear, crossbow, axe, and mace classes are not abstract. Healing potion, magic potion, tent, phoenix down classes inherit from the Item class because they are all types of items. Shield, helmet, chain mail, gauntlets, sword, spear, crossbow, axe, and mace classes are not abstract because they are specific types of items. The Item class is abstract because different types of items have different effects upon usage. Therefore, each must implement its own different types of effects. Armor and weapon are abstract because different types of weapons and armor give different types of stat boosts, special effect and set bonus. Each type must define the functions individually. All the inheritance is public.

Item abstract class
Armor abstract ->(inherits from) item class
Weapon abstract-> (inherits from) item class
Shield, helmet, chain mail, gauntlets (inherits from) armor class
Healing potion, magic potion, etc. (inherits from) item class
Sword, spear, crossbow, axe, mace(inherits from) weapon class

- The Player class is not abstract. It does not need inheritance since we can state strength, defense and experience in the private data member section. However, a player has transportation and items. We can assign maps for players. One map can store items as key and number of items the player has as value. A vector/map would be appropriate for storing transportations of the players depending on whether a player can have duplicates of the same transportation.

Each player “has a” inventory.
Each player “has a” transportation.