Q1.

a. **S.O.L.I.D principle violated is:** Dependency Inversion Principle

b. **Why**:

i. The high-level class Employer depends on the low-level classes HourlyWorker and SalaryWorker, if we added more worker types or more methods in the Employer class we'd have to do a lot of work in the Employer class.

b. Solution:

a. The high-level class Employer should depend on an interface with a calculatePay(hours) method. Then each class (HourlyWorker, SalaryWorker, any new worker types) should implement that interface.

b. Employer would have to change how it stores workers and how workers are added, there should be a single array of workers (an array of the interface type created in step i) and add different concrete types to it (i.e. keep the two loops in the constructor, but each loop adding to the same array).

Q2.

b. **S.O.L.I.D principle violated is**: Interface Segregation Principle

c. **Why**:

The **ILibraryItem** interface is overloaded with too many methods. A client should never be forced to implement an interface that it doesn't use, and clients shouldn't be forced to depend on methods they do not use.

c. **Solution**:

There's need to break the interface **ILibraryItem** into smaller interfaces.

We need to create two interfaces called:

i. **IBook** that has the **GetAuthor** and **GetTitle** method, then the Book Class can implement this interface.

ii. **IDVD** that has the **GetCastList** and **GetPlayTime**, **GetTitle** method, then the Book Class can implement this interface.

Q3.

i. **S.O.L.I.D principle violated is**: Single Responsibility Principle

ii. Why: The class ProfitReport does more than one functionality. This includes, SendToPrinter, SendToEmail, CreateReport. This violates the SRP, a class should only do one thing.

iii. Solution:

We need to create separate classes and delegate the SendToPrinter and SendToEmail responsibility to the class respectively.

i. PrintReport Class that the method SendToPrinter to delegate and handle printing jobs.

Ii. SendReportToEmail Class that have SendToEmail method that handles/sends report to emails.

Q4.

iv. **S.O.L.I.D principle violated is**: Liskov Substitution Principle

v. **Why**: The subtypeUSDAccount is not a member of BankAccount because it cannot provide the implementation of the behavior Credit and Debit.

vi. **Solution**: We need to create an interface IExchangeRate and the class BankAccount would implement the interface.

Q5:

a) **S.O.L.I.D principle violated is**: Dependency Inversion principle

b) It violates the principle because: CountryGDPReport being the higher module is made to depend on both lower modules Canada and Mexico respectively, but the principles states that the dependency should be on abstraction and not on concretions

Q6.

a. **S.O.L.I.D principle violated is:** Single Responsibility Principle
b. Why: The **PiggyBank** class if overloaded to many methods.
c. Solution: Create two separate class LoadCurrency and SaveCurrency to handle the load and save functionality of PiggyBank.

Q7.

a. **S.O.L.I.D principle violated is**: Interface Segregation Principle
b. **Why**:
   The **Iinsect** interface is overloaded too many methods. A client should never be forced to implement an interface that it doesn't use, however, clients shouldn't be forced to depend on methods they do not use.
c. Solution:
   There's a need to break the interface **Iinsect** into smaller interfaces. Thus, we need to create another interface called:
   IFlyingInsect that has the Fly method, then the **FlyingInsect** Class can implement this interface.
   IAquaticInsect that has the Swim method, then the AquaticInsect Class can implement this interface.