**11.9 *(Purchase Inheritance Hierarchy)*** Most manufacturing companies need to keep the account with their suppliers from whom they make purchase of raw materials. There are generally two types of purchase transaction, cash purchase and credit purchase. When an organization buys raw materials from suppliers for cash, it is called a cash purchase and when an organization purchases raw materials from the suppliers on credit, i.e., on condition of paying the amount at a specific later date, it is called a credit purchase. When a company makes a cash purchase, generally the suppliers give them a cash discount. On the other hand, when a company make a credit purchase, they may pay for some extra charges.

Create an inheritance hierarchy to represent various types of purchases made by manufacturing organizations, which might be used by a manufacturing company for keeping purchase information details.

Create class Purchase as the base class of the hierarchy, then include classes CashPurchase and CreditPurchase that derive from Purchase. Base class Purchase should include data members representing the name and id of the supplier, factory code of the factory for which the materials are procured, and item code and description of the material purchased, in addition to data members that store the quantity purchased in units and cost per unit. The constructor of the Purchase class should initialize these data members. The properties should validate the quantity purchased and the cost per unit, to ensure that the values are greater than or equal to 0.0; if not, throw an exception. The Purchase class should provide a public member function calculateCost that returns a double value indicating the cost associated with each purchase. Purchase’s calculateCost function should

determine the cost by multiplying the quantity by the cost per unit.

Derived class CashPurchase should inherit the functionality of base class Purchase, and also include a data member that represents a discount that is generally provided by the suppliers for cash purchase.A Discount property should check that the discount value is never less than zero, otherwise, it should throw an exception. Constructor of the CashPurchase class should receive a value to initialize the data members. The CashPurchase class should redefine member function calculateCost so that it computes the total cost by subtracting the discount from the quantity-based cost calculated by base class Purchase’s calculateCost function.

The derived class CreditPurchase should inherit directly from class Purchase and contain an additional data member representing an additional charge per unit charged for credit purchase. A CreditCharge property should check that the charges value is never less than zero, otherwise, it should throw an exception. The CreditPurchase class should redefine member function calculateCost so that it adds the additional charge per unit to the standard cost per unit before calculating the material cost.

Write a test app that creates objects of each type of Purchase and tests member function

calculateCost.