Question #1.

desiredAmount = user inputted amount at the ATM
accountBalance = current balance of user's account

If **accountBalance** <= 0
    Do not dispense money and tell the user their balance is negative or zero
Otherwise
    If **desiredAmount** > 500
        Do not dispense money and tell the user that withdrawals over 500$ are not allowed
Otherwise
    If **accountBalance** < **desiredAmount**
        Tells the user that their balance is not enough, but that the amount can still be withdrawn with a service charge of 25$, then ask for the user's input in confirming the withdrawal or not
            If the user confirms
                If **desiredAmount** > 300
                    Dispenses the money and perform
                    accountBalance = **accountBalance** - ((**desiredAmount** - 300) * 0.04) - (**desiredAmount** + 25)
                If **desiredAmount** <= 300
                    Dispenses the money and perform
                    accountBalance = accountBalance - (**desiredAmount** + 25)
            If the user denies
                Do not dispense money and end the operation
Otherwise
    If **accountBalance** >= **desiredAmount**
        If **desiredAmount** > 300
            Dispenses the money and perform
                accountBalance = **accountBalance** - ((**desiredAmount** - 300)*0.04) - **desiredAmount**
        If **desiredAmount** <= 300
            Dispenses the money and perform
                accountBalance = accountBalance - **desiredAmount**

It is important to note that the order of the checks matters. If the $500 check was made only at the end, the current last part of the code would be executed first and the desired result of the algorithm would not be achieved.

The question did not specify if the 4% service charge for amounts above $300 should be considered when checking if the account balance is enough for the desired amount. As an example, a $400 withdrawal actually costs $404 due to the 4% service charge on the $100 over $300, however, the current algorithm is not taking that into consideration for the last check. This, however, does not matter for the third check as the account is already expected to go into negative balance.

For completeness' sake, here is what the last check would look like with such considerations.

If **desiredAmount** > 300 AND **accountBalance** > 300 + ((**desiredAmount** - 300) * 0.04)
    Dispenses the money and perform
        accountBalance = accountBalance -
        ((**desiredAmount** - 300)*0.04) - **desiredAmount**
If **desiredAmount** <= 300
    Dispenses the money and perform
        accountBalance = accountBalance - **desiredAmount**

Question #2.

semesterCost = 25000
semesterUnits = 15
semesterWeeks = 15
cpscUnits = 4
cpscMeetingsPerWeek = 3
cpscHoursPerMeeting = 1

1- Find the cost per unit by dividing **semesterCost** by **semesterUnits**:
costPerUnit = **semesterCost** / **semesterUnits**
2- Find the cost of Computer Science by multiplying **costPerUnit** by **cpscUnits**:
cpscCostTotal = **costPerUnit** * **cpscUnits**
3- Find the cost per week by dividing **cpscCostTotal** by **semesterWeeks**
cpscCostWeekly = **cpscCostTotal** / **semesterWeeks**
4- Find the cost per meeting by dividing **cpscCostWeekly** by **cpscMeetingsPerWeek**
cpscCostMeeting = **cpscCostWeekly** / **cpscMeetingsPerWeek**
5- Find the cost per hour by dividing **cpscCostMeeting** by **cpscHoursPerMeeting**
cpscCostHour = **cpscCostMeeting** / **cpscHoursPerMeeting**

Using the given numbers we will have:

1- 25000 / 15 = $1,666.67 per unit
2- $1,666,67 * 4 = $6,666.67 total for the Computer Science Class
3- $6,666.67 / 15 = $444,44 per week of Computer Science Class
4- $444,44 / 3 = $148,15 per meeting of Computer Science Class
5- $148,15 / 1 = $148,15 per hour of Computer Science Class

Therefore, each hour of Computer Science Class costs the student $148,15. And since each class is one hour per meeting, each class also costs $148,15.

Question #3.

lawnArea = user inputted area of lawn to be mowed. Square yards
fertilizingQuantity = user inputted quantity of fertilizing applications
treesToPlant = user inputted quantity of trees to plant

1- Divide the lawn area by 5000, then multiply by $35
    mowingCost = (**lawnArea** / 5000) * 35
2- Multiply the fertilizing quantity by $30
    fertilizingCost = **fertilizingQuantity** * 30
3- Multiply the tree planting quantity by $50
    treesCost = **treesToPlant** * 50
4- Sum up all 3 costs to find total billing cost
    billingCost = **mowingCost** + **fertilizingCost** + **treesCost**

Running the algorithm with the following sample amounts: 25000 yards, 5 fertilizing applications and 10 trees planted:

1- 25000/5000 = 5
   5 * 35 = $175.00

2- 5 * 30 = $150.00

3- 10 * 50 = $500.00

4- 175 + 150 + 500 = $825.00 total billing cost