**Algorithm Development Task: Inventory Recording System**

* **Steps to design Algorithm**

 InputParsing

* Read the list of items with their details:
  + cen id (Item ID)
  + current stock (Current inventory)
  + rozecastes demana (Forecasted demand)
  + oreorder cost per urdt (Cost per unit)
  + batch size (Minimum units that can be ordered)

 Conditions

* If the current stock is less than the forecasted demand, calculate the deficit:

Deficit = Forecasted Demand – Current Stock

 BatchSize Adjustment

* If the deficit is greater than zero, order enough units to satisfy the deficit, but round up to the nearest batch size:

Unit to Order = [Deficit / Batch Size] \* Batch Size

 CostCalculation

* For each item, calculate the total reordering cost:

Total Cost = Unit to Order \* Recorder Cost per Unit

 Outputthe Plan

* Generate a list of reordering actions specifying cen id, units to order, and cost.

**C# Implementation for the Algorithm**

using System;

using System.Collections.Generic;

class InventoryReorderingSystem

{

**// Define a class to hold item details**

class Item

{

public string CenId { get; set; }

public int CurrentStock { get; set; }

public int ForecastedDemand { get; set; }

public decimal ReorderCostPerUnit { get; set; }

public int BatchSize { get; set; }

}

static void Main()

{

**// Sample data**

List<Item> items = new List<Item>

{

new Item { CenId = "Item1", CurrentStock = 50, ForecastedDemand = 120, ReorderCostPerUnit = 10.5m, BatchSize = 20 },

new Item { CenId = "Item2", CurrentStock = 30, ForecastedDemand = 70, ReorderCostPerUnit = 5.75m, BatchSize = 10 },

new Item { CenId = "Item3", CurrentStock = 100, ForecastedDemand = 80, ReorderCostPerUnit = 12.0m, BatchSize = 25 }

};

**// Generate reorder plan**

Console.WriteLine("Reordering Plan:");

foreach (var item in items)

{

int deficit = Math.Max(0, item.ForecastedDemand - item.CurrentStock);

int unitsToOrder = (int)Math.Ceiling((double)deficit / item.BatchSize) \* item.BatchSize;

decimal cost = unitsToOrder \* item.ReorderCostPerUnit;

**// Print the result**

Console.WriteLine($"Cen ID: {item.CenId}, Units to Order: {unitsToOrder}, Cost: {cost:C}");

}

}

}

**Flow Chart**

Start

Input: Item details(Cen ID, Current Stock, Demand, RecorderCostPerUnit, BatchSize)

End

Generate Recording Plan (ID, UnitToOrder, Cost)

For each item, check if CurrentStock < Forecasted Demand. If yes, calculate Deficit = Demand – Stock.

If Deficit > 0, calculate UnitToOrder:

UnitToOrder = Ceil(Deficit /

BatchSize) \* BatchSize

Calculate Cost: Cost = UnitToOrder \* RecorderCost

**INPUT**

Items:

1. CenID: Item1, CurrentStock: 50, Demand: 120, ReorderCost: 10.5, BatchSize: 20

2. CenID: Item2, CurrentStock: 30, Demand: 70, ReorderCost: 5.75, BatchSize: 10

3. CenID: Item3, CurrentStock: 100, Demand: 80, ReorderCost: 12.0, BatchSize: 25

**OUTPUT**

Reordering Plan:

Cen ID: Item1, Units to Order: 80, Cost: 840.00

Cen ID: Item2, Units to Order: 40, Cost: 230.00

Cen ID: Item3, Units to Order: 0, Cost: 0.00