**PRACTICE PROBLEMS**

**1. Working with XML in C#: Parsing and Manipulating XML**

**Scenario:** A company needs to update its product catalogue stored in XML format. The update involves adding new products, updating prices based on certain conditions, and removing products with low stock levels. Products have a stock level attribute that must be considered in the removal process.

**Sample Input (XML File - Products.xml):**

xml

<Products>

<Product>

<ID>1</ID>

<Name>Laptop</Name>

<Price>1200</Price>

<Stock>30</Stock>

</Product>

<Product>

<ID>2</ID>

<Name>Smartphone</Name>

<Price>800</Price>

<Stock>50</Stock>

</Product>

<Product>

<ID>3</ID>

<Name>Tablet</Name>

<Price>400</Price>

<Stock>5</Stock>

</Product>

</Products>

**Task:**

1. Add a new product with ID 4, Name "Smartwatch", Price 250, and Stock 15.
2. Update the price of the product with ID 2 to 850 if its stock is greater than 30.
3. Remove all products with stock levels less than 10.
4. Ensure that the remaining products are sorted by their ID in ascending order.

**Expected Output (Modified XML File):**

xml

<Products>

<Product>

<ID>1</ID>

<Name>Laptop</Name>

<Price>1200</Price>

<Stock>30</Stock>

</Product>

<Product>

<ID>2</ID>

<Name>Smartphone</Name>

<Price>850</Price>

<Stock>50</Stock>

</Product>

<Product>

<ID>4</ID>

<Name>Smartwatch</Name>

<Price>250</Price>

<Stock>15</Stock>

</Product>

</Products>

**2. Parsing CSV Files with a Generic Parser**

**Scenario:** A logistics company has a CSV file containing shipment records with additional columns for shipment date and delivery status. The analysis involves calculating total weight, average weight, highest weight shipment, and the percentage of on-time deliveries.

**Sample Input (CSV File - Shipments.csv):**

csv

ShipmentID,Origin,Destination,Weight,Date,Status

101,New York,Los Angeles,500,2024-08-01,On Time

102,Chicago,Miami,300,2024-08-02,Delayed

103,Dallas,Seattle,450,2024-08-01,On Time

104,San Francisco,New York,600,2024-08-03,On Time

105,Miami,San Francisco,350,2024-08-04,Delayed

**Task:**

1. Parse the CSV file and calculate the total weight of all shipments.
2. Calculate the average weight of shipments.
3. Determine the highest weight shipment and its details.
4. Calculate the percentage of shipments marked as "On Time."

**Expected Output:**

* Total Weight: 2200
* Average Weight: 440
* Highest Weight Shipment: ShipmentID: 104, Origin: San Francisco, Destination: New York, Weight: 600
* Percentage of On-Time Deliveries: 60%

**3. Excel Handling with CloseXML Library: Reading and Writing Excel Files**

**Scenario:** A financial analyst needs to generate a comprehensive summary of quarterly sales from an Excel file, including total sales, average sales, and growth rates compared to the previous quarter. The file also includes seasonal discounts that need to be factored into the calculations.

**Sample Input (Excel File - SalesData.xlsx):**

* **Sheet: Q1**

| **Product** | **Sales** | **Discount** | **Cost** |
| --- | --- | --- | --- |
| Laptop | 1500 | 100 | 1200 |
| Tablet | 800 | 50 | 600 |
| Smartphone | 1000 | 70 | 800 |

* **Sheet: Q2**

| **Product** | **Sales** | **Discount** | **Cost** |
| --- | --- | --- | --- |
| Laptop | 1700 | 120 | 1300 |
| Tablet | 900 | 60 | 700 |
| Smartphone | 1100 | 80 | 900 |

**Task:**

1. Calculate the total sales, average sales, and growth rates for each product over the two quarters.
2. Consider the discounts and costs to compute the net profit for each product.
3. Write the results to a new sheet named "Summary" including the percentage growth from Q1 to Q2.

**Expected Output (Excel File - SalesData.xlsx):**

* **Sheet: Summary**

| **Product** | **Total Sales** | **Average Sales** | **Growth Rate** | **Total Costs** | **Total Discounts** | **Net Profit** |
| --- | --- | --- | --- | --- | --- | --- |
| Laptop | 3200 | 1600 | 13.33% | 2500 | 220 | 680 |
| Tablet | 1700 | 850 | 12.50% | 1300 | 110 | 290 |
| Smartphone | 2100 | 1050 | 10.00% | 1700 | 150 | 250 |

**4. Working with JSON in C#: Parsing and Generating JSON**

**Scenario:** A company's HR system stores employee details in JSON format with nested arrays for projects and dependent objects for addresses. The system needs to handle updates such as adding new employees, updating details, and removing employees based on their project count. Additional constraints include handling promotions and ensuring data consistency across nested objects.

**Sample Input (JSON File - Employees.json):**

json

{

"Employees": [

{

"ID": "E001",

"Name": "John Doe",

"Department": "HR",

"Address": {

"Street": "123 Main St",

"City": "Springfield",

"State": "IL",

"Zip": "62701"

},

"Projects": ["Project A", "Project B"]

},

{

"ID": "E002",

"Name": "Jane Smith",

"Department": "Finance",

"Address": {

"Street": "456 Elm St",

"City": "Springfield",

"State": "IL",

"Zip": "62702"

},

"Projects": ["Project C"]

}

]

}

**Task:**

1. Add a new employee with ID E003, Name Alice Johnson, Department IT, Address including Street "789 Oak St", City "Springfield", State "IL", Zip "62703", and Projects ["Project D"].
2. Update the address of employee E001 to Street "321 Maple St" and ensure the Zip code is updated correctly.
3. Remove employees with fewer than 2 projects.
4. Add a new project "Project E" to all employees' project lists and ensure the list is sorted alphabetically.

**Expected Output (Modified JSON File):**

json

{

"Employees": [

{

"ID": "E001",

"Name": "John Doe",

"Department": "HR",

"Address": {

"Street": "321 Maple St",

"City": "Springfield",

"State": "IL",

"Zip": "62701"

},

"Projects": ["Project A", "Project B", "Project E"]

},

{

"ID": "E003",

"Name": "Alice Johnson",

"Department": "IT",

"Address": {

"Street": "789 Oak St",

"City": "Springfield",

"State": "IL",

"Zip": "62703"

},

"Projects": ["Project D", "Project E"]

}

]

}