Step-by-Step Implementation for Problem Statement based on Advanced Array Operations

1. ParseInput Method

```
static int[] ParseInput(string input)
{
    try
    {
        return input.Split(',')
                    .Select(s => int.Parse(s.Trim()))
                    .ToArray();
    }
    catch
    {
        return null;
    }
}
```

1. Splits the comma-separated string into individual strings
2. Trims whitespace from each element
3. Parses each string to integer
4. Returns null if parsing fails

2. CalculateMedian Method

```
static double CalculateMedian(int[] numbers)
{
```

```
    Array.Sort(numbers);

    int count = numbers.Length;



    if (count % 2 == 0)

    {

        return (numbers[count / 2 - 1] + numbers[count / 2]) / 2.0;

    }

    else

    {

        return numbers[count / 2];

    }
}
```

1. Sorts the array first
2. For even count: averages the two middle elements
3. For odd count: returns the middle element

3. FindSecondLargest Method

```
static string FindSecondLargest(int[] numbers)

{

    var uniqueNumbers = numbers.Distinct().OrderByDescending(n => n).ToArray();

    if (uniqueNumbers.Length < 2)

    {

        return "There is no second largest element.";

    }

    return uniqueNumbers[1].ToString();
}
```

1. Gets distinct values to handle duplicates
2. Sorts in descending order
3. Returns second element or error message

4. IsPalindrome Method

```
static bool IsPalindrome(int[] numbers)

{

    return numbers.SequenceEqual(numbers.Reverse());
}
```

1. Uses LINQ to compare array with its reverse
2. Returns true if they're identical

5. RotateArrayLeft Method

```
static void RotateArrayLeft(int[] numbers)

{

    if (numbers.Length > 0)

    {

        int firstElement = numbers[0];

        for (int i = 0; i < numbers.Length - 1; i++)

        {

            numbers[i] = numbers[i + 1];

        }

        numbers[numbers.Length - 1] = firstElement;

    }
}
```

1. Stores first element
2. Shifts all elements left by one position

3. Places stored first element at the end

6. Main Method Implementation

```
while (true)

{

    Console.WriteLine("\nMenu Options:");

    Console.WriteLine("1. Calculate Median");

    Console.WriteLine("2. Find Second Largest Element");

    Console.WriteLine("3. Check for Palindrome Array");

    Console.WriteLine("4. Rotate Array Left");

    Console.WriteLine("5. Exit");

    Console.Write("Choose an option: ");



    if (int.TryParse(Console.ReadLine(), out int choice))

    {

        switch (choice)

        {

            case 1:

                Console.WriteLine($"Median: {CalculateMedian(numbers)}");

                break;

            case 2:

                Console.WriteLine($"Second Largest Element:
{FindSecondLargest(numbers)}");

                break;

            case 3:
```

```
                Console.WriteLine($"Is Palindrome: {IsPalindrome(numbers)}");
                break;
            case 4:
                RotateArrayLeft(numbers);
                Console.WriteLine($"Array after rotation: {string.Join(", ",
numbers)}");
                break;
            case 5:
                Console.WriteLine("Exiting the program.");
                return;
            default:
                Console.WriteLine("Invalid option. Please select a valid menu
option.");
                break;
        }
    }
    else
    {
        Console.WriteLine("Invalid input. Please enter a number.");
    }
}
```

Complete Solution

```
using System;
using System.Linq;
```

```
class Program
{
    static void Main(string[] args)
    {
        int[] numbers = null;

        while (true)
        {
            Console.WriteLine("Please enter a list of integers
(comma-separated):");
            string input = Console.ReadLine();
            numbers = ParseInput(input);

            if (numbers != null)
            {
                break;
            }
            else
            {
                Console.WriteLine("Invalid input. Please enter a valid list of
integers.");
            }
        }
```

```
        while (true)
        {
            Console.WriteLine("\nMenu Options:");
            Console.WriteLine("1. Calculate Median");
            Console.WriteLine("2. Find Second Largest Element");
            Console.WriteLine("3. Check for Palindrome Array");
            Console.WriteLine("4. Rotate Array Left");
            Console.WriteLine("5. Exit");
            Console.Write("Choose an option: ");

            if (int.TryParse(Console.ReadLine(), out int choice))
            {
                switch (choice)
                {
                    case 1:
                        Console.WriteLine($"Median: {CalculateMedian(numbers)}");
                        break;
                    case 2:
                        Console.WriteLine($"Second Largest Element:
{FindSecondLargest(numbers)}");
                        break;
                    case 3:
                        Console.WriteLine($"Is Palindrome:
{IsPalindrome(numbers)}");
                        break;
```

```
                    case 4:
                        RotateArrayLeft(numbers);
                        Console.WriteLine($"Array after rotation: {string.Join(",
", numbers)}");
                        break;
                    case 5:
                        Console.WriteLine("Exiting the program.");
                        return;
                    default:
                        Console.WriteLine("Invalid option. Please select a valid
menu option.");
                        break;
                }
            }
            else
            {
                Console.WriteLine("Invalid input. Please enter a number.");
            }
        }
    }

    static int[] ParseInput(string input)
    {
        try
        {
```

```
        return input.Split(',')
                    .Select(s => int.Parse(s.Trim()))
                    .ToArray();
    }
    catch
    {
        return null;
    }
}

static double CalculateMedian(int[] numbers)
{
    Array.Sort(numbers);
    int count = numbers.Length;

    if (count % 2 == 0)
    {
        return (numbers[count / 2 - 1] + numbers[count / 2]) / 2.0;
    }
    else
    {
        return numbers[count / 2];
    }
}
```

```
    static string FindSecondLargest(int[] numbers)

    {

        var uniqueNumbers = numbers.Distinct().OrderByDescending(n =>
n).ToArray();

        if (uniqueNumbers.Length < 2)

        {

            return "There is no second largest element.";

        }

        return uniqueNumbers[1].ToString();

    }



    static bool IsPalindrome(int[] numbers)

    {

        return numbers.SequenceEqual(numbers.Reverse());

    }



    static void RotateArrayLeft(int[] numbers)

    {

        if (numbers.Length > 0)

        {

            int firstElement = numbers[0];

            for (int i = 0; i < numbers.Length - 1; i++)

            {
```

```
                numbers[i] = numbers[i + 1];
            }
            numbers[numbers.Length - 1] = firstElement;
        }
    }
}
```

Key Features

1. Input Validation: Proper error handling for invalid integer inputs
2. Menu System: Clear switch-case structure for different operations
3. Error Handling: Graceful handling of edge cases (like no second largest element)
4. Array Manipulation: Efficient algorithms for each operation
5. User-Friendly: Clear prompts and formatted output

The solution handles all the specified requirements including median calculation, second largest element finding, palindrome checking, and array rotation with proper input validation and error messages.