**1) c# code to make odd numbers from array to square**

Here's a simple C# program that takes an array of numbers and squares only the odd numbers:

using System;

using System.Linq;

class Program

{

static void Main()

{

int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

// Square only the odd numbers using LINQ

int[] squaredOdds = numbers.Select(n => n % 2 != 0 ? n \* n : n).ToArray();

Console.WriteLine("Squared odd numbers:");

Console.WriteLine(string.Join(", ", squaredOdds));

}

}

**Explanation:**

* The program defines an array of integers.
* It uses LINQ's .Select() method to iterate over each number.
* If the number is odd (n % 2 != 0), it squares the number; otherwise, it remains unchanged.
* The result is stored in a new array, squaredOdds, and printed to the console.

If you'd like to perform this operation without LINQ, you can use a simple loop:

using System;

class Program

{

static void Main()

{

int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

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

{

if (numbers[i] % 2 != 0)

{

numbers[i] \*= numbers[i]; // Square the odd number

}

}

Console.WriteLine("Squared odd numbers:");

Console.WriteLine(string.Join(", ", numbers));

}

}

Would you like to modify this further for a specific use case, such as filtering only squared odd numbers into a separate array? 🚀

Convert words to number

using System;

using System.Collections.Generic;

using System.Globalization;

public class WordToNumberConverter

{

// Dictionaries to hold word-to-number mappings.

private static readonly Dictionary<string, long> NumberTable = new Dictionary<string, long>(StringComparer.OrdinalIgnoreCase)

{

{ "zero", 0 },

{ "one", 1 },

{ "two", 2 },

{ "three", 3 },

{ "four", 4 },

{ "five", 5 },

{ "six", 6 },

{ "seven", 7 },

{ "eight", 8 },

{ "nine", 9 },

{ "ten", 10 },

{ "eleven", 11 },

{ "twelve", 12 },

{ "thirteen", 13 },

{ "fourteen", 14 },

{ "fifteen", 15 },

{ "sixteen", 16 },

{ "seventeen", 17 },

{ "eighteen", 18 },

{ "nineteen", 19 },

{ "twenty", 20 },

{ "thirty", 30 },

{ "forty", 40 },

{ "fifty", 50 },

{ "sixty", 60 },

{ "seventy", 70 },

{ "eighty", 80 },

{ "ninety", 90 }

};

private static readonly Dictionary<string, long> MultiplierTable = new Dictionary<string, long>(StringComparer.OrdinalIgnoreCase)

{

{ "hundred", 100 },

{ "thousand", 1000 },

{ "million", 1000000 },

{ "billion", 1000000000 },

{ "trillion", 1000000000000 }

};

/// <summary>

/// Converts a number in word form into its numeric representation.

/// </summary>

/// <param name="numberWords">The number in words (e.g., "one hundred twenty three").</param>

/// <returns>The numeric value as a long.</returns>

public static long ConvertWordsToNumber(string numberWords)

{

if (string.IsNullOrWhiteSpace(numberWords))

{

throw new ArgumentException("Input cannot be null or whitespace.");

}

// Normalize input by replacing hyphens and removing redundant words such as "and"

// This helps in handling inputs like "one hundred and twenty-three"

string normalized = numberWords.Replace("-", " ").Replace(" and ", " ");

string[] tokens = normalized.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

long current = 0;

long result = 0;

foreach (string token in tokens)

{

// If the token is a number word (e.g., "one", "twenty", "nine")

if (NumberTable.ContainsKey(token))

{

current += NumberTable[token];

}

// If it's the "hundred" multiplier, then multiply current by 100

else if (string.Equals(token, "hundred", StringComparison.OrdinalIgnoreCase))

{

current \*= MultiplierTable[token];

}

// For tokens like "thousand", "million", etc.

else if (MultiplierTable.ContainsKey(token))

{

current \*= MultiplierTable[token];

result += current;

current = 0; // reset for the next segment

}

else

{

throw new ArgumentException($"Invalid token encountered: {token}");

}

}

return result + current;

}

}

// Example usage:

public class Program

{

public static void Main()

{

string[] examples =

{

"one hundred twenty three",

"two thousand and fifty six",

"three hundred forty five thousand six hundred seventy eight",

"seven hundred"

};

foreach (var example in examples)

{

try

{

long result = WordToNumberConverter.ConvertWordsToNumber(example);

Console.WriteLine($"'{example}' => {result}");

}

catch (Exception ex)

{

Console.WriteLine($"Error processing '{example}': {ex.Message}");

}

}

}

}

Q]Find a number in generated random number, find the no. of occurrence to compare equal with input value.

using System;

using System.Linq;

namespace RandomNumberOccurrence

{

class Program

{

static void Main(string[] args)

{

// Step 1: Generate an array of random numbers.

int numberOfElements = 10; // Set the desired size of the array

int minValue = 1; // Minimum random number value (inclusive)

int maxValue = 100; // Maximum random number value (inclusive)

int[] randomNumbers = new int[numberOfElements];

Random random = new Random();

Console.WriteLine("Generated Random Numbers:");

for (int i = 0; i < numberOfElements; i++)

{

randomNumbers[i] = random.Next(minValue, maxValue + 1);

Console.Write(randomNumbers[i] + " ");

}

Console.WriteLine(); // Move to the next line

// Step 2: Get target value from user input.

Console.Write("Enter the number you want to search for: ");

string userInput = Console.ReadLine();

if (!int.TryParse(userInput, out int target))

{

Console.WriteLine("Invalid input. Please enter a valid integer.");

return;

}

// Step 3: Count the occurrences of the target number.

// You can use a loop or LINQ. Here's a LINQ example:

int occurrences = randomNumbers.Count(n => n == target);

Console.WriteLine($"The number {target} appears {occurrences} time(s) in the array.");

// Step 4: Compare if the occurrence count equals the target value.

if (occurrences == target)

{

Console.WriteLine("The number of occurrences is equal to the input value!");

}

else

{

Console.WriteLine("The number of occurrences is NOT equal to the input value.");

}

// Wait for user to close the console (optional)

Console.WriteLine("Press any key to exit.");

Console.ReadKey();

}

}

}

var words = new List<string> { "apple", "banana", "apple", "orange", "banana", "apple" };

output =apple

words.Select(x,y)

var mostFrequent = words

.GroupBy(w => w)

.OrderByDescending(g => g.Count())

.First()

.Key;

next question

input abcd#e

output edcb#a