

Dokumentimi I detyrave te shtepise.

Kapitulli 13

Introduction to Programming

Prof:Muzafer Shala

Ass:Laberion Zebica Student:Elion Krasniqi

Kampusi:FERIZAJ

**Exercises**

1. **Describe the strings in C#**. What is typical for the **string** type? Explain which the most important methods of the string class are.

**A string is a sequence of characters stored in a certain address in memory. Remember the type char? In the variable of type char we can record only one character. Where it is necessary to process more than one character then strings come to our aid.**

1. Write a program that reads a string, **reverse** it and prints it to the console. For example: "**introduction**"  "**noitcudortni**".

using System;

using System.Text;

public class Program

{

public static void Main()

{

string text = "introduction";

string reversed = ReverseText(text);

Console.WriteLine(reversed);

}

static string ReverseText(string text)

{

StringBuilder sb = new StringBuilder();

for (int i = text.Length - 1; i >= 0; i--)

{

sb.Append(text[i]);

}

return sb.ToString();

}

}

1. Write a program that **checks whether the parentheses are placed correctly** in an arithmetic expression. Example of expression with correctly placed brackets: **((a+b)/5-d)**. Example of an incorrect expression: **)(a+b))**.

using System;

using System.Collections.Generic;

public class BalancedBrackets {

    public class stack {

        public int top = -1;

        public char[] items = new char[100];

        public void push(char x)

        {

            if (top == 99)

            {

                Console.WriteLine("Stack full");

            }

            else {

                items[++top] = x;

            }

        }

        char pop()

        {

            if (top == -1)

            {

                Console.WriteLine("Underflow error");

                return '\0';

            }

            else

            {

                char element = items[top];

                top--;

                return element;

            }

        }

        Boolean isEmpty()

        {

            return (top == -1) ? true : false;

        }

    }

    static Boolean isMatchingPair(char character1,

                                  char character2)

    {

        if (character1 == '(' && character2 == ')')

            return true;

        else if (character1 == '{' && character2 == '}')

            return true;

        else if (character1 == '[' && character2 == ']')

            return true;

        else

            return false;

    }

    static Boolean areBracketsBalanced(char[] exp)

    {

        Stack<char> st = new Stack<char>();

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

        {

            if (exp[i] == '{' || exp[i] == '('

                || exp[i] == '[')

                st.Push(exp[i]);

            if (exp[i] == '}' || exp[i] == ')'

                || exp[i] == ']') {

                if (st.Count == 0)

                {

                    return false;

                }

                else if (!isMatchingPair(st.Pop(),

                                         exp[i])) {

                    return false;

                }

            }

        }

        if (st.Count == 0)

            return true; // balanced

        else

        {

            return false;

        }

    }

    public static void Main(String[] args)

    {

        char[] exp = { '{', '(', ')', '}', '[', ']' };

        if (areBracketsBalanced(exp))

            Console.WriteLine("Balanced ");

        else

            Console.WriteLine("Not Balanced ");

    }

}

4.    How many backslashes you must specify as an argument to the method **Split(…)** in order to **split the text by a backslash**?

Example: **one\two\three**.

Note: In C# backslash is an escaping character.

using System;

using System.Text;

namespace detyra4 {

class Program {

    static void Main(string[] args)

    {

        Console.WriteLine("Clean\\Evironment");

                Console.WriteLine("\\n\\t");

    }

}

}

5.    Write a program that detects how many times a substring is contained in the text. For example, let’s look for the substring "**in**" in the text:

|  |
| --- |
| We are liv**in**g **in** a yellow submar**in**e. We don't have anyth**in**g else. **In**side the submar**in**e is very tight. So we are dr**in**k**in**g all the day. We will move out of it **in** 5 days. |

The result is 9 occurrences.

using System;

public class Program

{

public static void Main()

{

string text = "We are living in a yellow submarine. We don't have anything else. Inside the submarine is very tight. So we are drinking all the day. We will move out of it in 5 days.";

string substring = Console.ReadLine();

Console.WriteLine("Substring \"{0}\" contains in the text {1} times.", substring, CountSubstrings(text, substring));

}

static int CountSubstrings(string text, string substring)

{

int count = 0;

int index = 0;

while (index != -1)

{

count = count + 1;

index = text.IndexOf(substring, index + 1);

}

return count;

}

}

6.    A text is given. Write a program that **modifies the casing** of letters to uppercase at all places in the text surrounded by **<upcase>** and **</upcase>** tags. Tags cannot be nested.

Example:

|  |
| --- |
| We are living in a <upcase>yellow submarine</upcase>. We don't have <upcase>anything</upcase> else. |

Result:

|  |
| --- |
| We are living in a YELLOW SUBMARINE. We don't have ANYTHING else. |

using System;

using System.Text;

class GFG{

    // Method to convert characters

    // of a string to opposite case

    static void convertOpposite(StringBuilder str)

    {

        int ln = str.Length;

        // Conversion according to ASCII values

        for (int i=0; i<ln; i++)

        {

            if (str[i]>='a' && str[i]<='z')

                //Convert lowercase to uppercase

                str[i] = (char)(str[i] - 32);

            else if(str[i]>='A' && str[i]<='Z')

                //Convert uppercase to lowercase

                str[i] = (char)(str[i] + 32);

        }

    }

    // Driver code

    public static void Main()

    {

        StringBuilder str = new StringBuilder("GeEkSfOrGeEkS");

        // Calling the Method

        convertOpposite(str);

        Console.WriteLine(str);

        }

}

7.    Write a program that reads a string from the console (20 characters maximum) and if shorter complements it right with "**\***" to 20 characters.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace PrintString

{

class PrintString

{

static void Main()

{

string input = Console.ReadLine();

StringBuilder text = new StringBuilder();

int maxSize = 20;

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

{

if (i < input.Length)

{

text.Append(input[i]);

}

else

{

text.Append('\*');

}

}

Console.WriteLine(text.ToString());

}

}

}

8.    Write a program that converts a given string into the form of array of Unicode escape sequences in the format used in the C# language. Sample input: "**Test**". Result: "**\u0054\u0065\u0073\u0074**".

using System;

using System.Text;

public class Program

{

public static void Main()

{

string given = "Test";

StringBuilder sb = new StringBuilder();

foreach (char character in given)

{

sb.Append(Convert.ToInt16(character) + " ");

}

Console.WriteLine(sb);

}

}

9.    Write a program that **encrypts a text** by applying XOR (excluding or) operation between the given source characters and given cipher code. The encryption should be done by applying XOR between the first letter of the text and the first letter of the code, the second letter of the text and the second letter of the code, etc. until the last letter of the code, then goes back to the first letter of the code and the next letter of the text. Print the result as a series of Unicode escape characters **\xxxx**.

using System;

public class XOREncryption

{

static String encryptDecrypt(String inputString)

{

char xorKey = 'P';

String outputString = "";

int len = inputString.Length;

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

{

outputString = outputString +

char.ToString((char) (inputString[i] ^ xorKey));

}

Console.WriteLine(outputString);

return outputString;

}

public static void Main(String[] args)

{

String sampleString = "Universum";

Console.WriteLine("Encrypted String");

String encryptedString = encryptDecrypt(sampleString);

Console.WriteLine("Decrypted String");

encryptDecrypt(encryptedString);

}

}

Sample source text: "**Test**". Sample cipher code: "**ab**". The result should be the following: "**\u0035\u0007\u0012\u0016**".

10.   Write a program that **extracts from a text all sentences that contain a particular word**. We accept that the sentences are separated from each other by the character "**.**" and the words are separated from one another by a character which is not a letter. Sample text:

|  |
| --- |
| We are living **in** a yellow submarine. We don't have anything else. Inside the submarine is very tight. So we are drinking all the day. We will move out of it **in** 5 days. |

Sample result:

|  |
| --- |
| We are living in a yellow submarine.  We will move out of it in 5 days. |

using System;

class GFG

{

public static void printWords(String s)

{

foreach (String val in s.Split(" "))

Console.WriteLine(val);

}

static public void Main()

{

Console.WriteLine("Jepni nje tekst");

String Str = Console.ReadLine();

printWords(Str);

}

}

11.   A string is given, composed of several **"forbidden" words** separated by commas. Also a text is given, containing those words. Write a program that **replaces the forbidden words with asterisks**. Sample text:

|  |
| --- |
| Microsoft announced its next generation C# compiler today. It uses advanced parser and special optimizer for the Microsoft CLR. |

Sample string containing the forbidden words: "**C#,CLR,Microsoft**".

Sample result:

|  |
| --- |
| \*\*\*\*\*\*\*\*\* announced its next generation \*\* compiler today. It uses advanced parser and special optimizer for the \*\*\*\*\*\*\*\*\* \*\*\*. |

using System;

using System.Text;

using System.Text.RegularExpressions;

public class Program

{

public static string StarReplacer(string text)

{

StringBuilder temp = new StringBuilder();

foreach (char letter in text)

{

temp.Append("\*");

}

string replaced = temp.ToString();

return replaced;

}

public static void Main()

{

string text = "Microsoft announced its next generation C# compiler today. It uses advanced parser and special optimizer for the Microsoft CLR.";

string forbidden = "C#,CLR,Microsoft";

char separator = ',';

string[] array = forbidden.Split(separator);

StringBuilder sb = new StringBuilder(text);

foreach (string word in array)

{

int index = 0;

string replacer = StarReplacer(word);

while (index != -1)

{

sb.Replace(word, replacer);

index = text.IndexOf(word+1);

}

}

Console.WriteLine(sb);

}

}

12.   Write a program that reads a number from console and prints it in **15-character field, aligned right** in several ways: as a decimal number, hexadecimal number, percentage, currency and exponential (scientific) notation.

using System;

using System.Threading;

class Number

{

static void Main(string[] args)

{

Thread.CurrentThread.CurrentCulture = new System.Globalization.CultureInfo("bg-BG");

int num = int.Parse(Console.ReadLine());

// decimal

Console.WriteLine("{0,15:D}", num);

// hexadecimal

Console.WriteLine("{0,15:X}", num);

// percent

Console.WriteLine("{0,15:P}", num);

// currency

Console.WriteLine("{0,15:C}", num);

// scientific notation

Console.WriteLine("{0,15:E}", num);

}

}

13.   Write a program that **parses an URL** in following format:

|  |
| --- |
| [protocol]://[server]/[resource] |

It should **extract** from the URL the protocol, server and resource parts. For example, when [**http://www.cnn.com/video**](http://www.cnn.com/video/) is passed, the result is:

|  |
| --- |
| [protocol]="http"  [server]="www.cnn.com"  [resource]="/video" |

using System;

using System.Text.RegularExpressions;

namespace URLAnalyzer

{

class Program

{

const string UrlPattern = @"\b(?<protocol>[a-zA-Z]\*)://(?<domain>[-a-zA-Z0-9.]+)(?<resource>/[-a-zA-Z0-9+&@#/%=~\_|!:,.;]\*)?";

static void Main(string[] args)

{

string url = Console.ReadLine();

Regex regex = new Regex(UrlPattern);

Match match = regex.Match(url);

if (!match.Success)

{

Console.WriteLine("ERROR");

}

else

{

string protocol = match.Groups["protocol"].ToString();

string domain = match.Groups["domain"].ToString();

string resource = match.Groups["resource"].ToString();

Console.WriteLine("[protocol]={0}", protocol);

Console.WriteLine("[domain]={0}", domain);

Console.WriteLine("[resource]={0}", resource);

}

}

}

}

14.   Write a program that **reverses the words in a given sentence** without changing punctuation and spaces. For example: "**C# is not C++ and PHP is not Delphi**"  "**Delphi not is PHP and C++ not is C#**".

using System;

using System.Linq;

using System.Text;

class ReverseSentance

{

static void Main(string[] args)

{

string sentance = Console.ReadLine(); ;

char[] punctuationChars = { ' ', '-', ',', ':', ';' };

StringBuilder result = new StringBuilder();

string[] words = sentance.Split(punctuationChars, StringSplitOptions.RemoveEmptyEntries);

for (int i = words.Length - 1; i >= 0; i--)

{

result.Append(words[i]);

// remove word from sentance

sentance = sentance.Remove(sentance.Length - words[i].Length);

// store punctuation chars between current word, and the next one

StringBuilder punctuaction = new StringBuilder();

while (sentance.Length > 0 && punctuationChars.Contains(sentance[sentance.Length - 1]))

{

punctuaction.Append(sentance[sentance.Length - 1]);

sentance = sentance.Remove(sentance.Length - 1);

}

// reverse punctuation chars, so we can keep puct. chars in original order

for (int j = punctuaction.Length - 1; j >= 0; j--)

{

result.Append(punctuaction[j]);

}

}

Console.WriteLine(result);

}

}

15.   A dictionary is given, which consists of several lines of text. Each line consists of a **word and its explanation**, separated by a hyphen:

|  |
| --- |
| .NET – platform for applications from Microsoft CLR – managed execution environment for .NET namespace – hierarchical organization of classes |

Write a program that **parses the dictionary** and then reads words from the console in a loop, **gives an explanation** for it or writes a message on the console that the word is not into the dictionary.

using System;

using System.Collections.Generic;

using System.Linq;

namespace Dictionary

{

class Program

{

private const string EndString = "End";

static void Main(string[] args)

{

int n = int.Parse(Console.ReadLine());

Dictionary<string, string> dictionary = new Dictionary<string, string>();

string currentLine = string.Empty;

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

{

currentLine = Console.ReadLine();

KeyValuePair<string, string> pair = Parse(currentLine);

if (!dictionary.ContainsKey(pair.Key))

{

dictionary.Add(pair.Key, pair.Value);

}

}

string currentCommand = Console.ReadLine();

while (currentCommand != EndString)

{

if (dictionary.ContainsKey(currentCommand))

{

Console.WriteLine("{0} - {1}", currentCommand, dictionary[currentCommand]);

}

else

{

Console.WriteLine("\"{0}\" not found!", currentCommand);

}

currentCommand = Console.ReadLine();

}

}

private static KeyValuePair<string, string> Parse(string currentLine)

{

string[] tokens = currentLine.Split('-');

string word = tokens[0].Trim();

string description = tokens[1].Trim();

KeyValuePair<string, string> pair = new KeyValuePair<string, string>(word, description);

return pair;

}

}

}

16.   Write a program that **replaces all hyperlinks** in a HTML document consisting of **<a href="…">…</a>** and hyperlinks in "forum" style, which look like **[URL=…]…[/URL]**.

Sample text:

|  |
| --- |
| <p>Please visit <a href="https://softuni.bg">our site</a> to choose a training course. Also visit <a href="www.devbg.org">our forum</a> to discuss the courses.</p> |

Sample result:

|  |
| --- |
| <p>Please visit [URL=https://softuni.bg]our site[/URL] to choose a training course. Also visit [URL=www.devbg.org]our forum[/URL] to discuss the courses.</p> |

using System;

using System.Text;

class HtmlTag

{

static void Main(string[] args)

{

string text = Console.ReadLine();

StringBuilder result = new StringBuilder();

int i = 0;

while (i >= 0 && text.Substring(i).Contains("<a href="))

{

// append the text before "<a href="

int lenght = text.IndexOf("<a href=", i) - i;

result.Append(text, i, lenght);

result.Append("[URL=");

i = text.IndexOf("<a href=", i);

lenght = text.IndexOf('>', i) - i - 10;

// append the url

result.Append(text, i + 9, lenght);

result.Append(']');

i = text.IndexOf('>', i) + 1;

}

result.Append(text.Substring(i));

result.Replace("</a>", "[/URL]");

Console.WriteLine(result);

}

}

17.   Write a program that **reads two dates** entered in the format "**day.month.year**" and calculates the **number of days between them**.

|  |
| --- |
| Enter the first date: 27.02.2006  Enter the second date: 3.03.2006  Distance: 4 days |

using System;

using System.Globalization;

namespace DateDistance

{

class DateDistance

{

static void Main(string[] args)

{

CultureInfo provider = CultureInfo.InvariantCulture;

string format = "dd.mm.yyyy";

string firstDateStr = Console.ReadLine();

string secondDateStr = Console.ReadLine();

try

{

DateTime firstDate = DateTime.ParseExact(firstDateStr, format, provider);

DateTime secondDate = DateTime.ParseExact(secondDateStr, format, provider);

TimeSpan distance = firstDate - secondDate;

Console.WriteLine(Math.Abs(distance.Days));

}

catch (FormatException)

{

Console.WriteLine("ERROR");

}

}

}

}

18.   Write a program that reads the date and time entered in the format "**day.month.year hour:minutes:seconds**" and prints the date and time after 6 hours and 30 minutes in the same format.

using System;

using System.Globalization;

class After6h30min

{

static void Main(string[] args)

{

string input = Console.ReadLine();

DateTime date = DateTime.ParseExact(input, "d.M.yyyy HH:mm:ss",

CultureInfo.InvariantCulture);

date = date.AddHours(6);

date = date.AddMinutes(30);

Console.WriteLine("{0:dd.MM.yyyy HH:mm:ss}", date);

}

}

19.   Write a program that **extracts all e-mail addresses** from a text. These are all substrings that are limited on both sides by text end or separator between words and match the shape **<sender>@<host>…<domain>**. Sample text:

|  |
| --- |
| Please contact us by phone (+001 222 222 222) or by email at example@gmail.com or at [test.user@yahoo.co.uk](mailto:test.user@yahoo.co.uk). This is not email: test@test. This also: @gmail.com. Neither this: a@a.b. |

Extracted e-mail addresses from the sample text:

|  |
| --- |
| example@gmail.com  test.user@yahoo.co.uk |

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

namespace EmailExtractor

{

class EmailExtractor

{

private static string[] ExtractEmails(string input)

{

string pattern = @"(\b[A-Z0-9.\_-]+)@[A-Z0-9][A-Z0-9.-]{0,61}[A-Z0-9]\.[A-Z.]{2,6}\b";

MatchCollection matches = Regex.Matches(input, pattern, RegexOptions.IgnoreCase);

string[] emails = new string[matches.Count];

int i = 0;

foreach (var match in matches)

{

emails[i] = match.ToString();

i++;

}

return emails;

}

static void Main(string[] args)

{

string text = Console.ReadLine();

string[] emails = ExtractEmails(text);

foreach (string email in emails)

{

Console.WriteLine(email);

}

}

}

}

20.   Write a program that **extracts from a text all dates** written in format **DD.MM.YYYY**and prints them on the console in the standard format for Canada. Sample text:

|  |
| --- |
| I was born at 14.06.1980. My sister was born at 3.7.1984. In 5/1999 I graduated my high school. The law says (see section 7.3.12) that we are allowed to do this (section 7.4.2.9). |

Extracted dates from the sample text:

|  |
| --- |
| 14.06.1980  3.7.1984 |

using System;

using System.Globalization;

using System.Text.RegularExpressions;

using System.Threading;

using System.Text;

class Dates

{

static void Main(string[] args)

{

string text = Console.ReadLine();

Regex dateRegex = new Regex(@"(0?[1-9]|[12][0-9]|3[01])[.](0?[1-9]|1[012])[.]\d{4}");

MatchCollection dates = dateRegex.Matches(text);

Thread.CurrentThread.CurrentCulture = new CultureInfo("en-CA");

StringBuilder output = new StringBuilder();

foreach (var date in dates)

{

output.AppendLine(date.ToString());

}

Console.WriteLine(output);

}

}

21.   Write a program that extracts from a text all words which are **palindromes**, such as **ABBA**", "**lamal**", "**exe**".

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace PalindromeExtractor

{

class PalindromeExtractor

{

private static bool IsPalindrome(string word)

{

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

{

if (word[i] != word[word.Length - i - 1])

{

return false;

}

}

return true;

}

static void Main(string[] args)

{

string text = Console.ReadLine();

char[] separators = { '.', ',', ' ', '!', '?', '-', '\_' };

string[] words = text.Split(separators, StringSplitOptions.RemoveEmptyEntries);

foreach (string word in words)

{

if (IsPalindrome(word))

{

Console.WriteLine(word);

}

}

}

}

}

22.   Write a program that reads a string from the console and prints in alphabetical order **all letters from the input string and how many times each one of them occurs** in the string.

using System;

class LettersCount

{

static void Main(string[] args)

{

string text = Console.ReadLine();

int[] lettersCount = new int[52];

foreach (var c in text)

{

if (c >= 'a' && c <= 'z')

{

int index = (c - 'a') \* 2;

lettersCount[index]++;

}

else if (c >= 'A' && c <= 'Z')

{

int index = ((c - 'A') \* 2) + 1;

lettersCount[index]++;

}

}

bool upCase = false;

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

{

if (lettersCount[i] > 0)

{

char charA = upCase ? 'A' : 'a';

char letter = (char)(i / 2 + charA);

Console.WriteLine("{0}: {1}", letter, lettersCount[i]);

}

upCase = !upCase;

}

}

}

23.   Write a program that reads a string from the console and prints in alphabetical order **all words from the input string** and **how many times each one of them occurs** in the string.

using System;

using System.Collections.Generic;

namespace WordCounter

{

class WordCounter

{

static void Main(string[] args)

{

string text = Console.ReadLine();

char[] separators = { ' ', ',', '.', '?', '!' };

string[] wordArray = text.Split(separators, StringSplitOptions.RemoveEmptyEntries);

SortedDictionary<string, int> words = new SortedDictionary<string, int>();

foreach (string word in wordArray)

{

if (words.ContainsKey(word))

{

words[word]++;

}

else

{

words.Add(word, 1);

}

}

foreach (var word in words)

{

Console.WriteLine("{0} -> {1}", word.Key, word.Value);

}

}

}

}

24.   Write a program that reads a string from the console and replaces every sequence of identical letters in it with a single letter (the **repeating** letter). Example: "**aaaaabbbbbcdddeeeedssaa**"  "**abcdedsa**".

using System;

using System.Text;

class ConsecutiveLetters

{

static void Main(string[] args)

{

string input = Console.ReadLine();

StringBuilder result = new StringBuilder(input);

for (int i = 1; i < result.Length; i++)

{

bool isLatinLetter =

result[i] >= 'a' && result[i] <= 'z' ||

result[i] >= 'A' && result[i] <= 'Z';

if (result[i] == result[i - 1] && isLatinLetter)

{

result.Remove(i, 1);

i--;

}

}

Console.WriteLine(result);

}

}

25.   Write a program that reads a list of words separated by commas from the console and prints them in alphabetical order (after **sorting**).

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace WordSorter

{

class WordSorter

{

private static string[] separators = { "," };

static void Main(string[] args)

{

string str = Console.ReadLine();

string[] words = str.Split(separators, StringSplitOptions.RemoveEmptyEntries);

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

{

words[i] = words[i].Trim();

}

Array.Sort(words);

foreach (string word in words)

{

Console.WriteLine(word);

}

}

}

}

26.   Write a program that **extracts all the text without any tags and attribute values** from an HTML document.

Sample text:

|  |
| --- |
| <html>      <head><title>News</title></head>      <body><p><a href="https://softuni.bg">Telerik          Academy</a>aims to provide free real-world practical          training for young people who want to turn into          skillful software engineers.</p></body>  </html> |

Sample result:

|  |
| --- |
| News  Telerik Academy aims to provide free real-world practical training for young people who want to turn into skillful software engineers. |

using System;

using System.Text;

using System.Text.RegularExpressions;

class HtmlTagRemove

{

static void Main(string[] args)

{

const string TITLE\_OPEN\_TAG = "<title";

const string TITLE\_CLOSE\_TAG = "</title>";

StringBuilder input = new StringBuilder();

string line;

do

{

line = Console.ReadLine();

input.Append(line.Trim() + ' ');

}

while (line != "</html>");

StringBuilder output = new StringBuilder();

// title

int indexOfTitle = input.ToString().IndexOf(TITLE\_OPEN\_TAG);

int startIndex = input.ToString().IndexOf('>', indexOfTitle) + 1;

int lenght = input.ToString().IndexOf(TITLE\_CLOSE\_TAG) - startIndex;

string title = input.ToString().Substring(startIndex, lenght);

input = input.Remove(0, startIndex + lenght);

output.AppendLine("Title: " + title);

// body

Regex tag = new Regex(@"<(?![!/]?[ABIU][>\s])[^>]\*>");

string body = tag.Replace(input.ToString(), "");

output.AppendLine("Body:");

output.Append(body.Trim());

Console.WriteLine(output);

}

}