**1.**

Составить регулярное выражение, определяющее является ли заданная строка IP адресом, записанным в десятичном виде.

– пример правильных выражений: 127.0.0.1, 255.255.255.0.

– пример неправильных выражений: 1300.6.7.8, abc.def.gha.bcd.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace task1

{

public static class StringExtension

{

public static string RegexIp(this string text)

{

Regex reg = new Regex(@"^(\d|[1-9]\d|1\d\d|2([0-4]\d|5[0-5]))\.(\d|[1-9]\d|1\d\d|2([0-4]\d|5[0-5]))\.(\d|[1-9]\d|1\d\d|2([0-4]\d|5[0-5]))\.(\d|[1-9]\d|1\d\d|2([0-4]\d|5[0-5]))$");

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

return (mat.Value);

}

return "0";

}

}

class Program

{

static void Main(string[] args)

{

string text1 = "127.0.0.1", text2 = "255.255.255.0", text3 = "1300.6.7.8", text4 = "abc.def.gha.bcd";

string t1 = text1.RegexIp();

string t2 = text2.RegexIp();

string t3 = text3.RegexIp();

string t4 = text4.RegexIp();

Console.WriteLine(t1);

Console.WriteLine(t2);

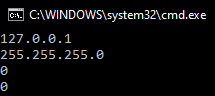
Console.WriteLine(t3);

Console.WriteLine(t4);

}

}

}



**2.** Написать регулярное выражение, определяющее является ли данная строка GUID с или без скобок. Где GUID это строчка, состоящая из 8, 4, 4, 4, 12 шестнадцатеричных цифр разделенных тире.

– пример правильных выражений: e02fd0e4-00fd-090A-ca30-0d00a0038ba0.

– пример неправильных выражений: e02fd0e400fd090Aca300d00a0038ba0.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace task2

{

public static class StringExtension

{

public static string RegexMac(this string text)

{

Regex reg = new Regex(@"(?i)\{?[a-f\d]{8}-([a-f\d]{4}-){3}[a-f\d]{12}\}?", RegexOptions.IgnorePatternWhitespace);

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

return (mat.Value);

}

return "0";

}

}

class Program

{

static void Main(string[] args)

{

string text1 = "e02fd0e4-00fd-090A-ca30-0d00a0038ba0", text2 = "e02fd0e400fd090Aca300d00a0038ba0";

string m = text1.RegexMac();

string a = text2.RegexMac();

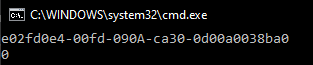
Console.WriteLine(m);

Console.WriteLine(a);

}

}

}



**3.** Написать регулярное выражение, определяющее является ли данная строчка валидным URL адресом. В данной задаче правильным URL считаются адреса http и https, явное указание протокола также может отсутствовать. Учитываются только адреса, состоящие из символов, т.е. IP адреса в качестве URL не присутствуют при проверке. Допускаются поддомены, указание порта доступа через двоеточие, GET запросы с передачей параметров, доступ к подпапкам на домене, допускается наличие якоря через решетку. Однобуквенные домены считаются запрещенными. Запрещены спецсимволы, например «–» в начале и конце имени домена. Запрещен символ «\_» и пробел в имени домена. При составлении регулярного выражения ориентируйтесь на список правильных и неправильных выражений заданных ниже.

– пример правильных выражений: <http://www.example.com>, http://example.com.

– пример неправильных выражений: Just Text, <http://a.com>.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace task3

{

public static class StringExtension

{

public static string RegexUrl(this string text)

{

Regex reg = new Regex(@"^(https?:\/\/)?([\w\.]{3,})\.([a-z]{2,6}\.?)(\/[\w\.]\*)\*\/?$");

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

return (mat.Value);

}

return "0";

}

}

class Program

{

static void Main(string[] args)

{

string text1 = "http://www.example.com", text2 = "http://example.com", text3 = "Just Text", text4 = "http://a.com";

string m = text1.RegexUrl();

string a = text2.RegexUrl();

string i = text3.RegexUrl();

string l = text4.RegexUrl();

Console.WriteLine(m);

Console.WriteLine(a);

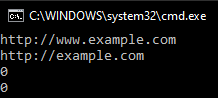
Console.WriteLine(i);

Console.WriteLine(l);

}

}

}



**4.** Написать регулярное выражение, определяющее является ли данная строчка датой в формате dd/mm/yyyy. Начиная с 1600 года до 9999 года.

– пример правильных выражений: 29/02/2000, 30/04/2003, 01/01/2003.

– пример неправильных выражений: 29/02/2001, 30-04-2003, 1/1/1899.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace task4

{

public static class StringExtension

{

public static string RegexDate(this string text)

{

Regex reg = new Regex(@"((0[1-9]|1\d|2[0-8])/(0[1-9]|1[0-2])/\d{4})|((29/(0[1-9]|1[0-2])/ ([02468][048]00|[13579][26]00|\d\d0[48]|\d\d[2468][048]|\d\d[13579][26])

)|(29/(0[13-9]|1[0-2])/\d{3}[1235679]))|(30/(0[13-9]|1[0-2])/\d{4})|(31/(0[13578]|10|12)/\d{4})", RegexOptions.IgnorePatternWhitespace);

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

return (mat.Value);

}

return "0";

}

}

class Program

{

static void Main(string[] args)

{

string text1 = "29/02/2000", text2 = "30/04/2003", text3 = "01/01/2003", text4 = "29/02/2001", text5 = "30-04-2003", text6 = "1/1/1899";

string d = text1.RegexDate();

string a = text2.RegexDate();

string t = text3.RegexDate();

string e = text4.RegexDate();

string m = text5.RegexDate();

string c = text6.RegexDate();

Console.WriteLine(d);

Console.WriteLine(a);

Console.WriteLine(t);

Console.WriteLine(e);

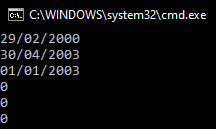
Console.WriteLine(m);

Console.WriteLine(c);

}

}

}



**5.** Есть текст с ценами в рублях. Извлечь из него цены.

– пример правильных выражений: 23.78 руб.

– пример неправильных выражений: 22 р., 0.002 руб.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace task5

{

public static class StringExtension

{

public static string RegexPercent(this string text)

{

Regex reg = new Regex(@"(^[1-9]+|^[0-9]+\.{1,1}[0-9]{1,2})\sруб\.", RegexOptions.IgnorePatternWhitespace);

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

return (mat.Value);

}

return "0";

}

}

class Program

{

static void Main(string[] args)

{

string text = "20.78 руб.", text2 = "22 р., 0.002 руб.";

string r = text.RegexPercent();

string u = text2.RegexPercent();

Console.WriteLine(r);

Console.WriteLine(u);

}

}

}



**6.** Проверить, надежно ли составлен пароль. Пароль считается надежным, если он состоит из 8 или более символов. Где символом может быть английская буква, цифра и знак подчеркивания. Пароль должен содержать хотя бы одну заглавную букву, одну маленькую букву и одну цифру.

– пример правильных выражений: C00l\_Pass, SupperPas1.

– пример неправильных выражений: Cool\_pass, C00l.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

namespace task6

{

public static class StringExtension

{

public static string RegexPassword(this string text)

{

Regex reg = new Regex(@"^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])(?!.\*\s).{8,}$$", RegexOptions.IgnorePatternWhitespace);

MatchCollection matches = reg.Matches(text);

foreach (Match mat in matches)

{

return (mat.Value);

}

return "0";

}

}

class Program

{

static void Main(string[] args)

{

string text1 = "C00l\_Pass", text2 = "SupperPas1", text3 = "Cool\_pass", text4 = "C00l";

string p = text1.RegexPassword();

string a = text2.RegexPassword();

string s = text3.RegexPassword();

string w = text4.RegexPassword();

Console.WriteLine(p);

Console.WriteLine(a);

Console.WriteLine(s);

Console.WriteLine(w);

}

}

}

