**Лёгкий уровень**

**Задание № 1**

static void Exam()

{

string scoreString = Console.ReadLine();

string scoreName = Console.ReadLine();

List<string> passedStudents = FilterPassedtudents(scoreString, scoreName);

if (passedStudents.Count > 0)

{

foreach (string student in passedStudents)

{

Console.WriteLine($"{student}");

}

}

else Console.WriteLine("Никто");

static List<string> FilterPassedtudents(string scoreString, string scoreName)

{

List<string> passed = new List<string>();

string[] scores = scoreString.Split(',');

string[] names = scoreName.Split(',');

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

{

if (Convert.ToInt32(scores[i]) >= 35)

{

passed.Add(names[i]);

}

}

return passed;

}

}

**Задание № 2**

static void Geometry()

{

//string sideString = Console.ReadLine().Trim();

string sideString = "5 1 4";

string answer = isRightTriangle(sideString) ? "Да" : "Нет";

Console.WriteLine(answer);

static bool isRightTriangle(string sideString)

{

string[] sidesStr = sideString.Split(' ');

int[] sides = new int[sidesStr.Length];

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

{

sides[i] = Convert.ToInt32(sidesStr[i]);

}

Array.Sort(sides);

Array.Reverse(sides);

if (Math.Pow(sides[0], 2) == Math.Pow(sides[1], 2) + Math.Pow(sides[2], 2)) return true;

else return false;

}

}

**Задание № 3**

static void SymbolsHiding()

{

//string stringToMask = Console.ReadLine();

string stringToMask = "78";

string result = Mask(stringToMask);

Console.WriteLine(result);

static string Mask(string stringToMask)

{

string newString = string.Empty;

if (stringToMask.Length > 4)

{

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

{

if (i < stringToMask.Length - 4) newString += "#";

else newString += stringToMask[i];

}

}

else newString = stringToMask;

return newString;

}

}

**Средний уровень**

**Задание № 1**

static void Survey()

{

string inpupString = Console.ReadLine();

int result = DigitalRoot(int.Parse(inpupString));

Console.WriteLine(result);

static int DigitalRoot(int n)

{

int result = n;

int vremResult = 0;

do

{

vremResult = 0;

for (int i = 0; i < result.ToString().Length; i++) // result = 18

{

vremResult += Convert.ToInt32(result.ToString()[i].ToString());

}

result = vremResult;

} while (vremResult >= 10);

return result;

}

}

**Задание № 2**

static void Raiting()

{

Dictionary<string, int[]> candidates = GetCandidates();

foreach (string candidate in SelectedCandidates(candidates)) Console.WriteLine(candidate);

static Dictionary<string, int[]> GetCandidates()

{

Dictionary<string, int[]> dict = new Dictionary<string, int[]>();

string candidate;

while ((candidate = Console.ReadLine()) != null && candidate != "")

{

var parts = candidate.Split(',');

string name = parts[0];

var scores = parts.Skip(1).Select(int.Parse).ToArray();

dict[name] = scores;

}

return dict;

}

static List<string> SelectedCandidates(Dictionary<string, int[]> candidates)

{

List<string> result = new List<string>();

List<Human> cands = new List<Human>();

ICollection<string> keys = candidates.Keys;

foreach (string candidate in keys)

{

int[] numbers = candidates[candidate];

double average = Math.Round(numbers.Average(), 1);

if (average >= 5) cands.Add(new Human(candidate, average));

}

cands.Sort();

System.Globalization.CultureInfo.CurrentCulture = System.Globalization.CultureInfo. GetCultureInfo("en-US"); // или using System.Globalization;

foreach (Human h in cands) result.Add($"{h.Name},{h.Rating:F1}");

return result;

}

}

class Human : IComparable<Human>

{

public string Name { get; set; }

public double Rating { get; set; }

public Human(string name, double rating)

{

Name = name;

Rating = rating;

}

public int CompareTo(Human obj)

{

int result = -Rating.CompareTo(obj.Rating);

if (result == 0) result = Name.CompareTo(obj.Name);

return result;

}

}

**Задание № 3**

static void DominantNumbers()

{

string stockPrices = Console.ReadLine();

string dominatePrices = GetdominatePrices(stockPrices);

Console.WriteLine(dominatePrices);

static string GetdominatePrices(string stockPrices)

{

List<int> numbers = stockPrices.Split(' ').Select(Int32.Parse).ToList();

string choosen = string.Empty;

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

{

int current = numbers[i];

if (checkMax(i, numbers)) choosen += current + " ";

numbers.RemoveAt(i);

i--;

}

choosen = choosen.TrimEnd();

return choosen;

}

static bool checkMax(int a, List<int> lst)

{

bool result = true;

foreach (int i in lst)

{

if (lst[a] < i) result = false;

}

return result;

}

}

**Задание № 4**

static void checkingPasswords()

{

string inputString = Console.ReadLine();

StringAnalyzer analyzer = new StringAnalyzer(inputString);

int count = analyzer.CountWordsWithDigits();

if (count == 0) Console.WriteLine("Не обнаружено");

else Console.WriteLine(count);

}

class StringAnalyzer

{

private string text;

public StringAnalyzer(string text)

{

this.text = text;

}

public int CountWordsWithDigits()

{

string[] words = text.Split(' ');

var result = from p in words where char.IsUpper(p[0]) && p.Any(char.IsDigit) select p;

return result.Count();

}

}

**Сложный уровень**

**Задание № 1**

static void ReportMaker()

{

string inputData = Console.ReadLine();

List<string> report = GenerateMonthlyReport(inputData);

foreach (string line in report) Console.WriteLine(line);

static List<string> GenerateMonthlyReport(string data)

{

List<string> result = new List<string>();

List<Item> products = new List<Item>();

string[] sales = data.Split(';');

foreach (string sale in sales)

{

string[] dateNameAmount = sale.Split(":");

Item item = new Item(dateNameAmount[0], dateNameAmount[1], dateNameAmount[2]);

products.Add(item);

}

products = products.OrderBy(p => p.MonthNumber).ThenBy(p => p.Product).ToList();

string currentMonth = "";

foreach (Item i in products)

{

if (i.Month != currentMonth)

{

currentMonth = i.Month;

result.Add(i.Month + ":");

}

result.Add($"-{i.Product}:{i.Quantity}:{i.MonthNumber}");

}

return result;

}

}

class Item

{

public string Date { get; set; }

public string Product { get; set; }

public int Quantity { get; set; }

public string Month

{

get

{

var months = new Dictionary<string, string>

{

{ "01", "Январь" },

{ "02", "Февраль" },

{ "03", "Март" },

{ "04", "Апрель" },

{ "05", "Май" },

{ "06", "Июнь" },

{ "07", "Июль" },

{ "08", "Август" },

{ "09", "Сентябрь" },

{ "10", "Октябрь" },

{ "11", "Ноябрь" },

{ "12", "Декабрь" }

};

string m = Date.Split('-')[1];

return months.ContainsKey(m) ? months[m] : "Неизвестный месяц";

}

}

public int MonthNumber

{

get

{

return int.Parse(this.Date.Split('-')[1]);

}

}

public Item(string date, string product, string quantity)

{

Date = date;

Product = product;

Quantity = int.Parse(quantity);

}

}

**Задание № 2**

static void Backlogs()

{

string studentsInfo = Console.ReadLine();

string scoresInfo = Console.ReadLine();

Dictionary<string, List<int>> courseScores = new Dictionary<string, List<int>>();

var studentRecords = studentsInfo.Split(new char[] { ';' }, StringSplitOptions.RemoveEmptyEntries);

foreach (var record in studentRecords)

{

var parts = record.Split(',');

string studentName = parts[0];

string course = parts[1];

int score = int.Parse(parts[2]);

if (!courseScores.ContainsKey(course))

courseScores[course] = new List<int>();

courseScores[course].Add(score);

}

Dictionary<string, int> passingScores = new Dictionary<string, int>();

List<string> courseOrder = new List<string>();

var scoreRecords = scoresInfo.Split(new char[] { ';' }, StringSplitOptions.RemoveEmptyEntries);

foreach (var scoreRecord in scoreRecords)

{

var parts = scoreRecord.Split(',');

string course = parts[0];

int passing = int.Parse(parts[1]);

passingScores[course] = passing;

courseOrder.Add(course);

}

List<string> result = GetCoursesWithoutDebts(courseScores, passingScores, courseOrder);

if (result.Count == 0) Console.WriteLine("Пусто");

else

{

foreach (var course in result) Console.WriteLine(course);

}

static List<string> GetCoursesWithoutDebts(Dictionary<string, List<int>> courseScores, Dictionary<string, int> passingScores, List<string> couseOrder)

{

List<string> list = new List<string>();

foreach (var subject in couseOrder)

{

if (courseScores[subject].Min() > passingScores[subject])

{

list.Add(subject);

}

}

return list;

}

}

**Задание № 3**

static void Lottery()

{

string inputString = Console.ReadLine();

//string inputString = "153";

string result = NextSmaller(inputString);

Console.WriteLine(result);

static string NextSmaller(string n)

{

string[] combinations = makeSequences(n); //создаём комбинации из заданных циферек

int ind = Array.IndexOf(combinations, n); //индекс элемента в массиве

if (ind == 0) return "-1";

else if (combinations[ind - 1][0] == '0') return "-1"; //если предыдущий элемент начинается с нуля

else return combinations[ind - 1]; //предыдущее значение

}

static string[] makeSequences(string numbers)

{

//нужно определить количество уникальных символов, чтобы понять количество комбинаций

Dictionary<char, int> dic = new Dictionary<char, int>();

foreach (char ch in numbers)

{

if (dic.ContainsKey(ch)) dic[ch]++;

else dic[ch] = 1;

}

int N = fac(numbers.Length); //общее количество комбинаций это факториал числа неповторяющихся символов (числитель формулы)

int D = 1; //считаем знаменатель формулы перестановки с повторами

foreach (var e in dic) D = D \* fac(e.Value);

int amount = N / D; //формула перестановки с повторяющимися элементами

string[] combinations = new string[amount]; //список комбинаций

int n = 0; //счётчик добавленных в массив строк

Random rnd = new Random(); //класс для случайных чисел

while (n < amount) //пока не заполним массив

{

numbers = new string(numbers.OrderBy(x => rnd.Next()).ToArray<char>()); //создаём случайную последовательность из заданных цифр

if (!combinations.Contains(numbers))

{

combinations[n] = numbers; //добасляем в массив

n++;

}

}

combinations = combinations.Order().ToArray<string>();

return combinations;

}

static int fac(int x) //факториал числа

{

int amount = 1;

for (int i = 1; i < x + 1; i++) amount \*= i;

return amount;

}

}

**Задание № 4**

static void Coding() //ЗАДАЧА НЕ РЕШЕНА (НЕ СОВПАДАЕТ ОТВЕТ)!

{

string inputString = Console.ReadLine();

string encodedString = EncodeString(inputString);

Console.WriteLine(encodedString);

static string EncodeString(string input)

{

//этот метод предоставляется условием и создаёт 34 последовательных числа

List<string> GenerateBinaryNumbers(int n)

{

var binaryNumbers = new List<string>();

for (int i = 1; i <= n; i++)

{

binaryNumbers.Add(Convert.ToString(i, 2));

}

return binaryNumbers;

}

string text = input.ToLower();

List<string> bnums = GenerateBinaryNumbers(34); //получаем 34 кода для символов в порядке возрастаня сложности

foreach (string s in bnums) Console.WriteLine(s); //вывести коды по порядку

Dictionary<char, int> symbols = new Dictionary<char, int>(); //сохраняем сиволы строки в словарь

foreach (char ch in text)

{

if (!symbols.ContainsKey(ch)) symbols.Add(ch, 1); else symbols[ch] += 1;

}

symbols = (from p in symbols orderby p.Value ascending, p.Key descending select p).Reverse().ToDictionary(); //словарь сортируем по частоте символов

foreach (KeyValuePair<char, int> p in symbols) Console.WriteLine($"{p.Key}:{p.Value}"); //вывести элементы словаря символов с частотой

Dictionary<char, string> dic = new Dictionary<char, string>();

for (int i = 0; i < symbols.Count; i++) dic.Add(symbols.ElementAt(i).Key, bnums[i]);

foreach (KeyValuePair<char, string> p in dic) Console.WriteLine($"{p.Key} : {p.Value}"); //вывести символы и соответствующие коды

string coded = string.Empty;

foreach (char ch in text) coded += dic[ch];

return coded;

}

}

**Задание № 5**

static void Users\_access()

{

var userManager = new UserManager();

var inputLines = new List<string>();

string? line;

while ((line = Console.ReadLine()) != null)

{

if (line == "") break;

inputLines.Add(line);

}

foreach (var command in inputLines)

{

string[] com = command.Split(' ');

switch (com[0])

{

case "add\_user":

userManager.AddUser(com[1], Convert.ToInt32(com[2]));

break;

case "remove\_user":

userManager.RemoveUser(com[1]);

break;

case "promote":

userManager.Promote(com[1]);

break;

case "demote":

userManager.Demote(com[1]);

break;

case "get\_users":

userManager.GetUsers();

break;

}

}

}

public class UserManager

{

private Dictionary<string, int> users = new Dictionary<string, int>();

public UserManager AddUser(string name, int level = 1)

{

users.Add(name, level);

return this;

}

public UserManager RemoveUser(string name)

{

users.Remove(name);

return this;

}

public UserManager Promote(string name)

{

users[name] += 1;

return this;

}

public UserManager Demote(string name)

{

if(users[name] > 0)

{

users[name] -= 1;

}

return this;

}

public void GetUsers()

{

if(users.Count > 0)

{

foreach (var user in users)

{

Console.WriteLine($"{user.Key}: {user.Value}");

}

}

else Console.WriteLine("Не найдено");

}

}