|  |  |
| --- | --- |
| для прик эмбл | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  Высшего образования "МИРЭА-Российский технологический университет" | |
| Институт комплексной безопасности и специального приборостроения. | |
| Кафедра КБ-3 «Управление и моделирование систем» | |
|  |  |
|  |
|  |
| **Отчет** | |
| **По домашней работе по дисциплине «Теория автоматов и формальных языков»** | |
|  | |

**Студент**: Гусев Дмитрий Викторович

**Группа**: БСБО-09-17

**Номер студенческого билета**: 17Б0111

**Отчет по домашней работе принял: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Корягин С.В.)**

**Домашнюю работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Гусев Д.В.)**

*Подпись обучающегося* *Ф.И.О. исполнителя*

# Задание 1

**Вариант № 7**

**Постановка задания**

Написать программу, которая переводит исходный текст на английском языке в число арабского и римского формата с подробной диагностикой и выводом ошибки пользователю.

**Примеры работы программы**

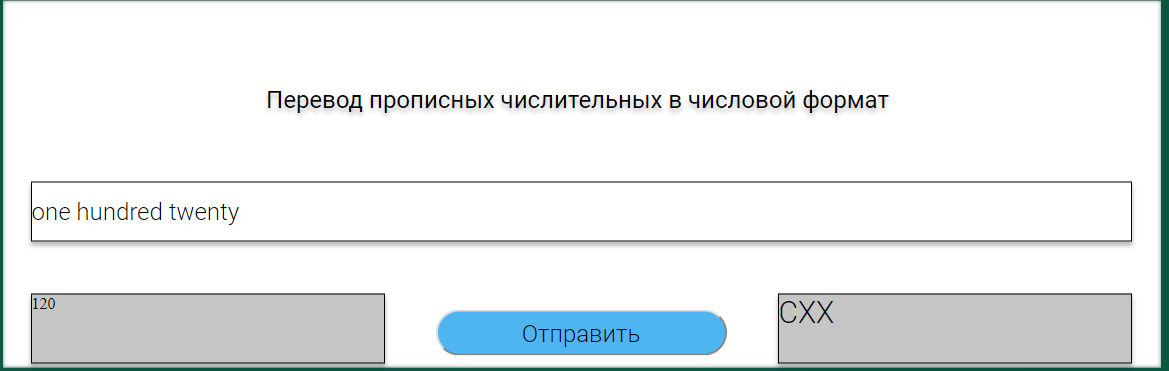


Рисунок 1. Правильный исходный текст

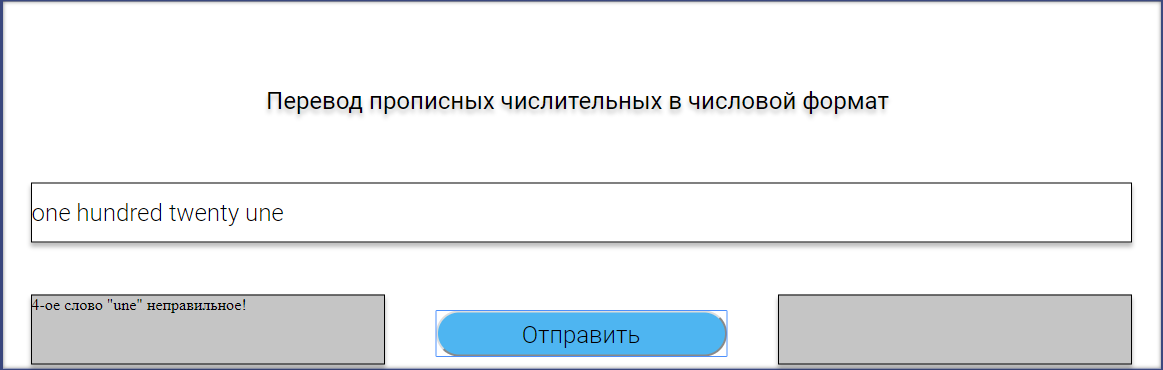


Рисунок 2. Неправильное слово

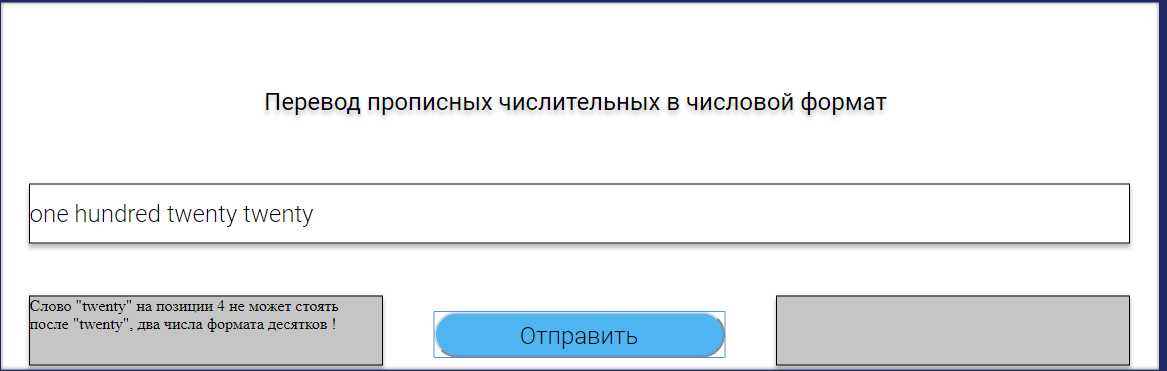


Рисунок 3. Два числа формата десятков

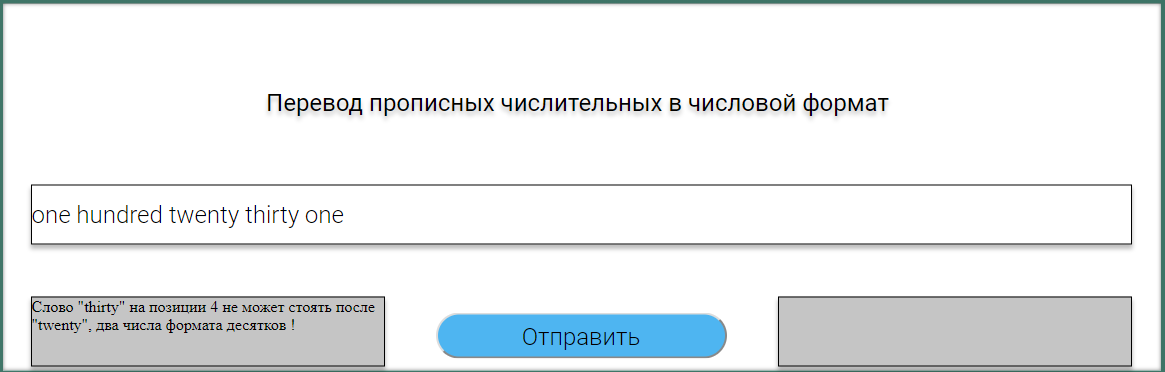


Рисунок 4. Два число формата десятков

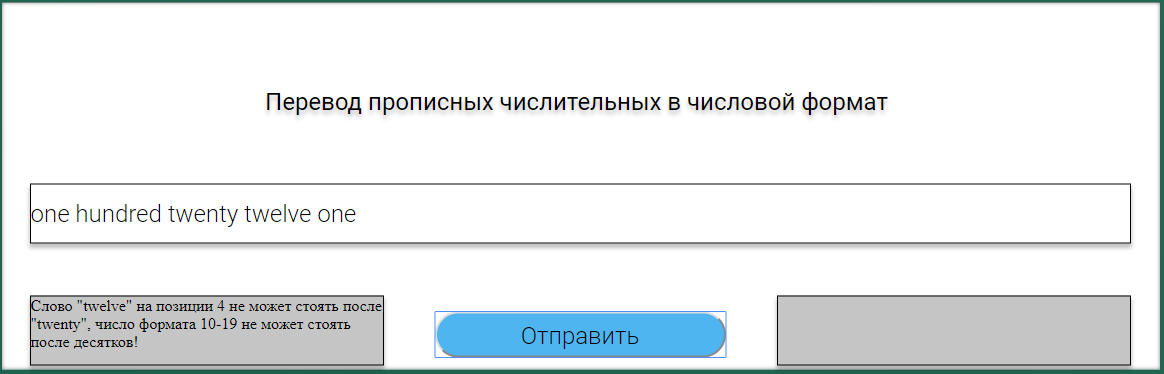


Рисунок 5. Число 10-19 после десятков

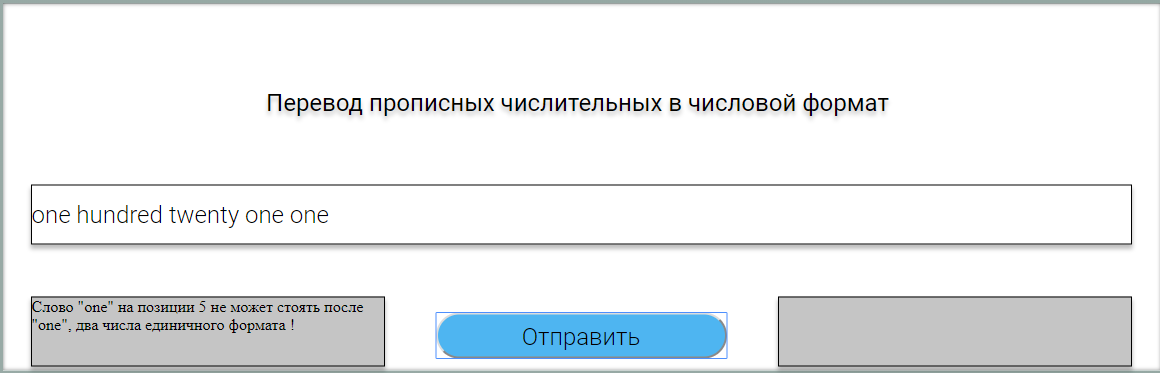


Рисунок 6. Два числа единичного формата

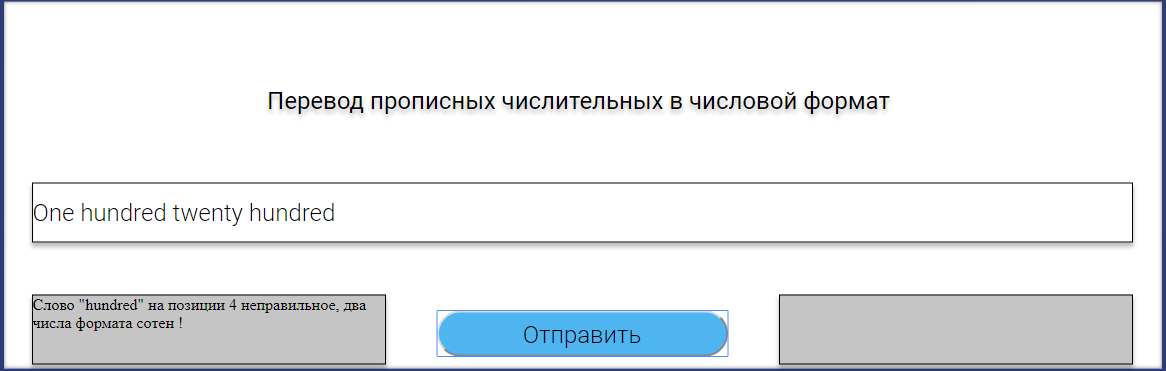


Рисунок 7. Две сотни

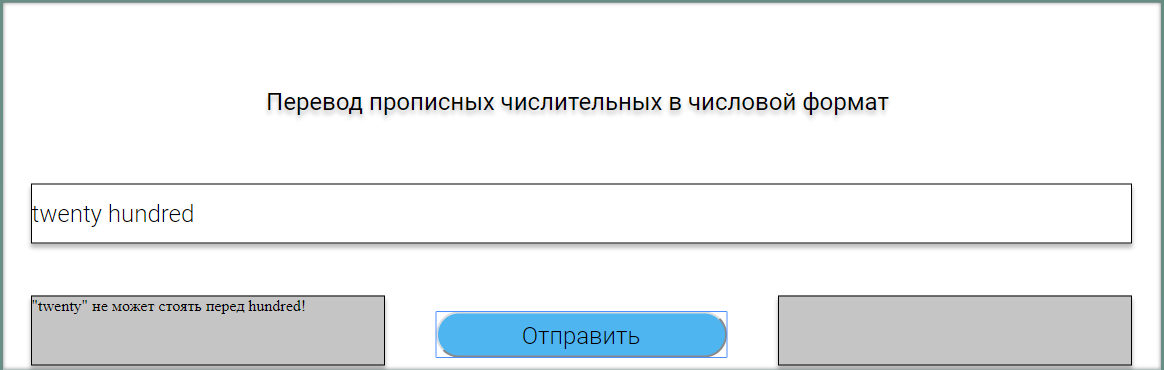


Рисунок 8. Десятки перед сотнями

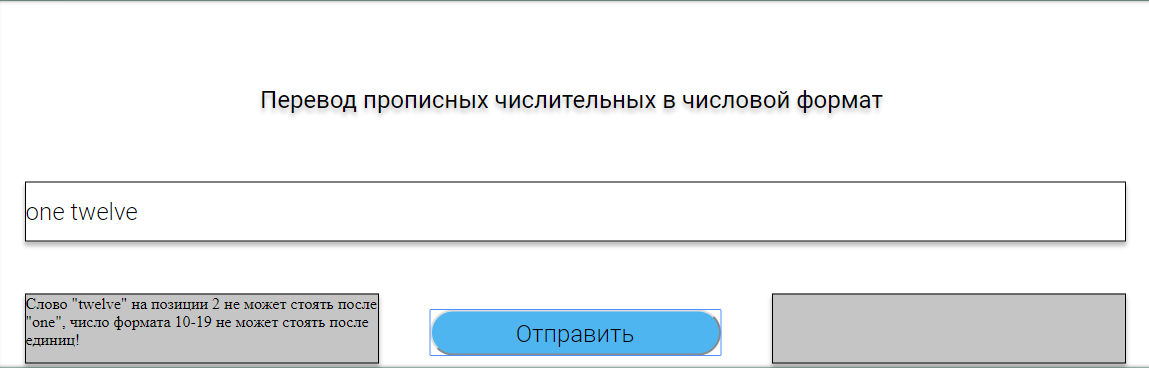


Рисунок 9. 10-19 после единиц

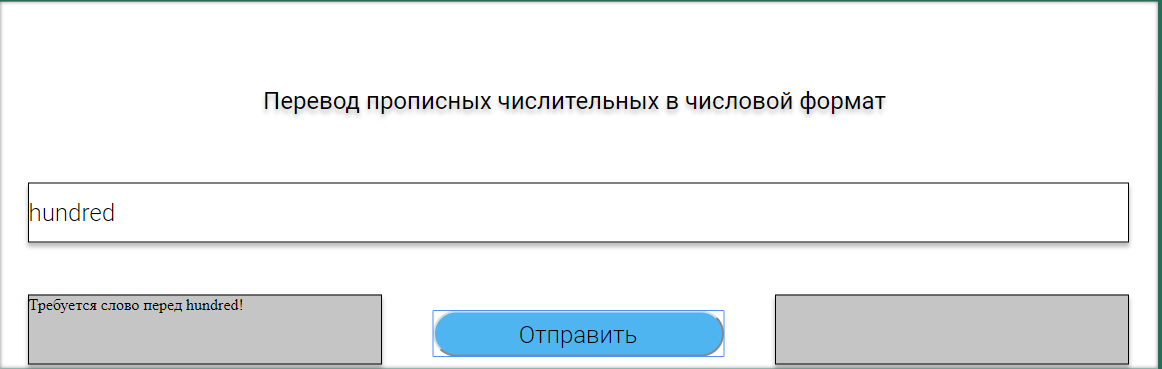


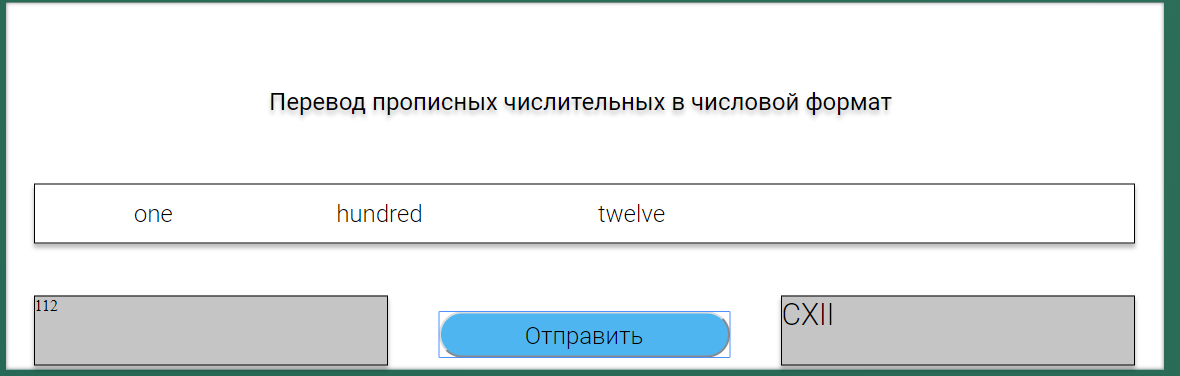
Рисунок 10. Перед сотней требуется слово

Рисунок 11. Корректная работа программы

# Задание 2

**Вариант № 3**

**Постановка задания**

Разработать программы, которая перемешивает две строки, поступающие на вход в шахматном порядке. Слова поочередно берутся вначале из первой строки, потом из второй.

**Примеры работы программы**

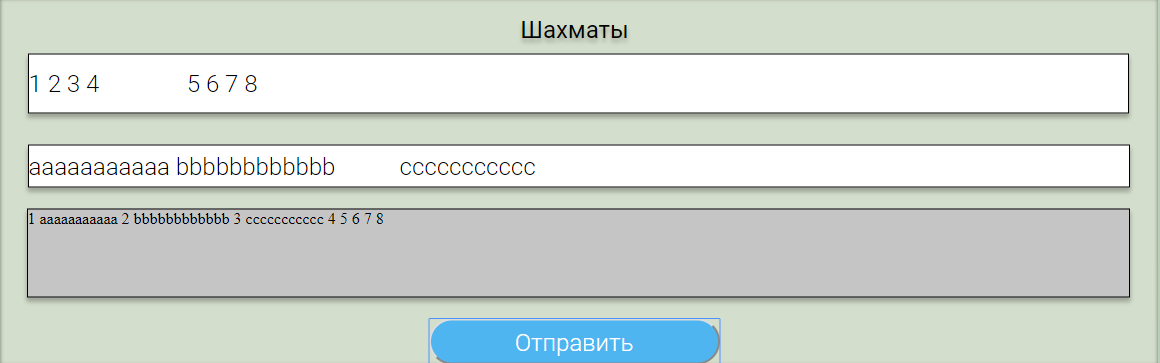


Рисунок 12. Пример 1

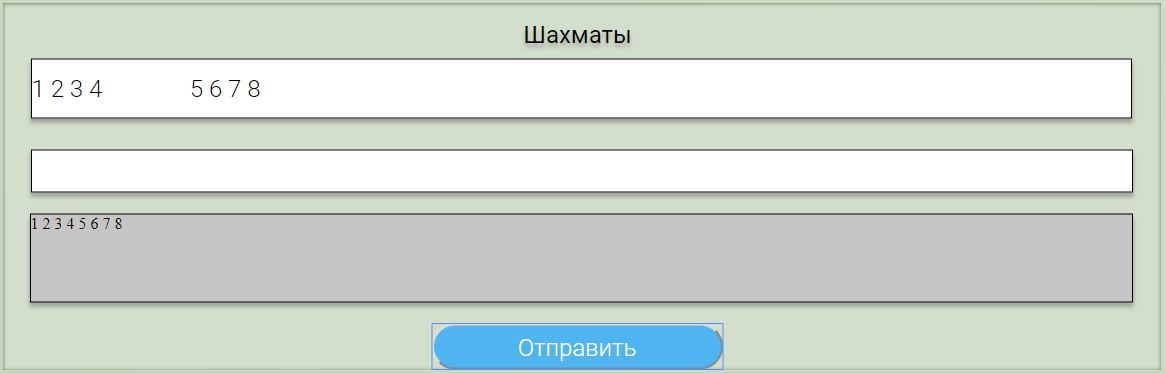


Рисунок 13. Пустая входная строка

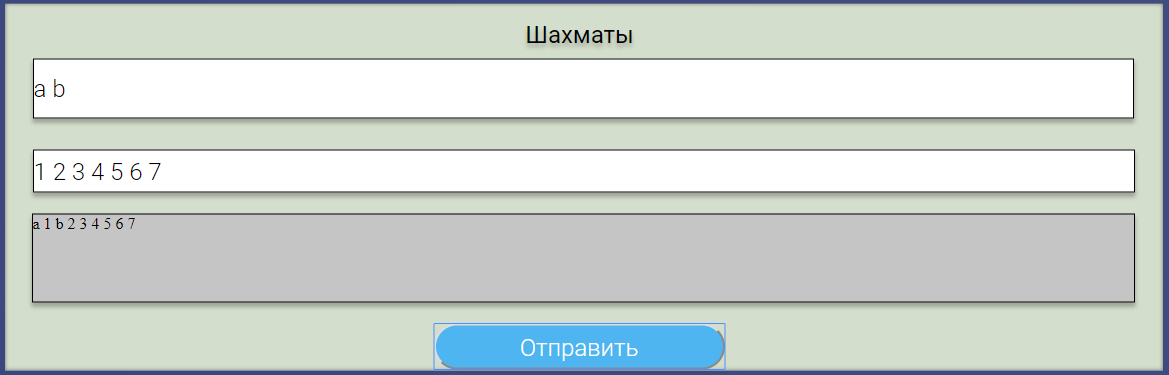


Рисунок 14. Первая строка короче второй

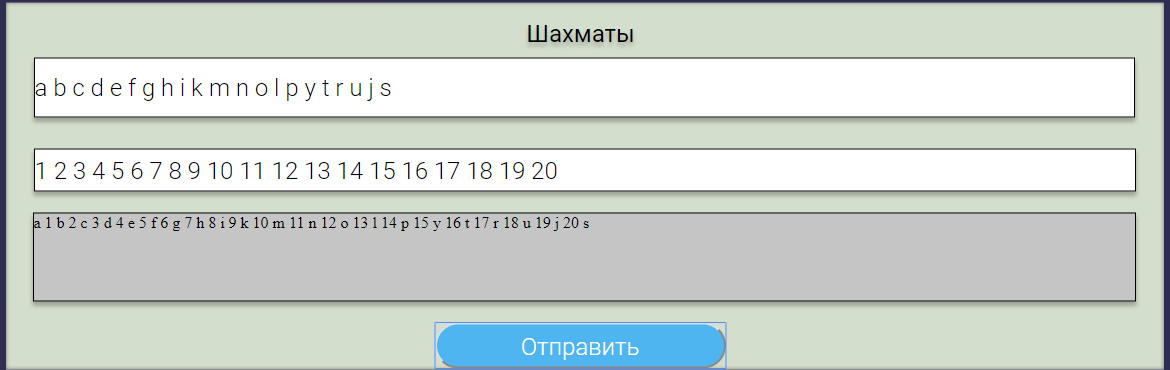


Рисунок 15. Равные по размеру строки

# Листинг

Задание 1

**Script.js**

|  |
| --- |
| function getNumbers() {  let outRoman = 0;  let outArabian = "";  let url = "http://91.240.86.250:9092/api/numeral";  fetch(url, {  method: 'POST', // \*GET, POST, PUT, DELETE, etc.  mode: 'cors', // no-cors, cors, \*same-origin  headers: {  'Content-Type': 'application/json',  // 'Content-Type': 'application/x-www-form-urlencoded',  },  body: JSON.stringify(document.getElementById("input").value), // body data type must match "Content-Type" header  })  .then(response => response.json()).then(json => {  outArabian = json.num; console.log(outArabian);  outRoman = json.romanian;  errorCode = json.errorCode;  errorWord = json.errorWord;  sentence = json.fullSentence;  ranks = json.ranks  if (errorCode == 0) {  document.getElementById("arabian").innerHTML = (outArabian === 0 ? "" : outArabian);  document.getElementById("roman").innerHTML = outRoman;  }  else {  let errorMessage = getErrorMessage(errorCode, errorWord, sentence, ranks);  document.getElementById("arabian").innerHTML = errorMessage;  document.getElementById("roman").innerHTML = "";  }  console.log(errorCode);  console.log(outRoman);  console.log(json);  });  }  function getErrorMessage(errorCode = 0, errorWord = "", sentence = "", ranks = []) {  if (errorCode > 1000 && errorCode < 2000) {  let pos = Math.floor(errorCode % 1200 / 10) + 1;  let rank = errorCode % 10;  let message = "";  console.log(pos);  console.log(rank);  switch (rank) {  case 1:  message = ", единицы";  break;  case 2:  message = ", десятки";  break;  case 3:  message = ", два числа формата сотен";  break;  case 4:  message = ", число формата 10-19";  break;  default:  message = "Error";  break;  }  let postmessage = "";  if(ranks[3] == true)  {  if(rank == 4)  {  message = ", два числа формата 10-19";  }  else  {  postmessage = "не может стоять после числа формата 10-19"  }  }  else if(ranks[0] == true)  {  if(rank == 1)  {  message = ", два числа единичного формата";  }  else  {  postmessage = "не может стоять после единиц"  }  }  else if(ranks[1] == true)  {  if(rank == 2)  {  message = ", два числа формата десятков";  }  else  {  postmessage = "не может стоять после десятков"  }  }  if(rank == 3)  {  postmessage = "";  return `Слово \"${errorWord}\" на позиции ${pos} неправильное${message} ${postmessage}!`;  }    return `Слово \"${errorWord}\" на позиции ${pos} не может стоять после \"${sentence}\"${message} ${postmessage}!`;  }  if (errorCode > 219 && errorCode < 230) {  let pos = Math.floor(errorCode % 220) + 1;  return `${pos}-ое слово \"${errorWord}\" неправильное!`;  }  if (errorCode > 29 && errorCode < 40) {  let pos = Math.floor(errorCode % 30) + 1;  return `${pos}-ое слово \"${errorWord}\" неправильное!`;  }  switch (errorCode) {  case 11:  return `\"${errorWord}\" не может стоять перед hundred!`;  break;  case 21:  return `Слишком большое число слов дано, уберите слово \"${errorWord}\"!`;  break;  case 23:  return `Неправильное слово \"${errorWord}\" на 2-ой позиции, возможно имелось ввиду hundred!`;  break;  case 40:  return `Требуется слово перед hundred!`;  break;  default:  return `Неизвестная ошибка!`;  break;  }  } |

Handler.cs

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.IO;  using System.Linq;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  using Automata\_theory.Model;  using Microsoft.Extensions.Options;  namespace Automata\_theory.Lib  {  public interface IHandler  {  List<Numeral> GetNumerals(string filePath);  Number GetNumberDecimal(string line);  Number GetNumberRomanian(Number number);  ChessLine GetShuffledLine(ChessLine chessLine);  }  public class Handler : IHandler  {  public static List<Numeral> Numerals { get; set; } = new List<Numeral>();  public Handler(IOptions<Settings> settings)  {  GetNumerals(Environment.CurrentDirectory + settings.Value.numeralsPath);  }  public List<Numeral> GetNumerals(string filePath)  {  List<string> file = File.ReadAllLines(filePath).ToList();  Numerals = (from line in file  let split = line.Split(" ")  where int.TryParse(split[0], out \_) && int.TryParse(split[2], out \_) && split.Count() > 2  select new Numeral() { Numerical = int.Parse(split[0]), Word = split[1], Rank = short.Parse(split[2]) }).ToList();  return Numerals;  }    public Number GetNumberDecimal(string line)  {  Number number = new Number();  if (line.Equals(string.Empty))  return number;  line = line.ToLower();  string lineTrimmed = Regex.Replace(line, @"\s\s+", " ").Trim();  List<string> split = lineTrimmed.Split(" ").ToList();  number.FullSentence = lineTrimmed;  for (int i = 0; i < split.Count; i++)  {  Numeral numeral = Numerals.Find(x => x.Word.Equals(split[i]));  if (numeral == null)  {  number.ErrorCode = 220 + i;  number.ErrorWord = split[i];  return number;  }  if (numeral.Word.Equals("hundred") && i == 0)  {  number.ErrorCode = 40;  number.ErrorWord = split[i];  return number;  }  number.Add(numeral, i);  if (number.ErrorCode != 0)  {  if (number.ErrorCode > 1200)  {  int rank = number.ErrorCode % 10;  switch (rank)  {  case 1:  {  if (number.ranks[2])  {  number.FullSentence = string.Concat((from item in split.Skip(1)  let n = Numerals.Find(x => x.Word.Equals(item))  where n.Rank == 1 || n.Rank == 4  select item + " ").Take(1)).Trim();  break;  }  number.FullSentence = (from item in split  let n = Numerals.Find(x => x.Word.Equals(item))  where n.Rank == 1 || n.Rank == 4  select item).Take(1).First();  break;  }  case 2:  {  if (number.ranks[2])  {  number.FullSentence = (from item in split.Skip(1)  let n = Numerals.Find(x => x.Word.Equals(item))  where n.Rank == 1 || n.Rank == 2 || n.Rank == 4  select item).Take(1).First();  break;  }  number.FullSentence = (from item in split  let n = Numerals.Find(x => x.Word.Equals(item))  where n.Rank == 1 || n.Rank == 2 || n.Rank == 4  select item).Take(1).First();  break;  }  case 3:  {  number.FullSentence = (from item in split  let n = Numerals.Find(x => x.Word.Equals(item))  where n.Rank == 3  select item).Take(1).First();  break;  }  case 4:  {  if (number.ranks[3])  {  number.FullSentence = (from item in split  let n = Numerals.Find(x => x.Word.Equals(item))  where n.Rank == 4  select item).Take(1).First();  break;  }  if (number.ranks[0] && number.ranks[1])  {  if ((number.ErrorCode % 1200 / 10) < 2)  {  number.FullSentence = (from item in split  let n = Numerals.Find(x => x.Word.Equals(item))  where n.Rank == 1  select item).Take(1).First();  break;  }  if (number.ranks[2])  {  number.FullSentence = string.Concat((from item in split.Skip(1)  let n = Numerals.Find(x => x.Word.Equals(item))  where n.Rank == 1 || n.Rank == 2  select item + " ").Take(2)).Trim();  break;  }  number.FullSentence = string.Concat((from item in split  let n = Numerals.Find(x => x.Word.Equals(item))  where n.Rank == 1 || n.Rank == 2  select item + " ").Take(2)).Trim();  break;  }  if (number.ranks[2])  {  number.FullSentence = string.Concat((from item in split.Skip(1)  let n = Numerals.Find(x => x.Word.Equals(item))  where n.Rank == 1 || n.Rank == 2  select item + " ").Take(1)).Trim();  break;  }  number.FullSentence = (from item in split  let n = Numerals.Find(x => x.Word.Equals(item))  where n.Rank == 1 || n.Rank == 2  select item).Take(1).First();  break;  }  default:  break;  }  }  return number;  }  }  if (split.Count > 2 && !split[1].Equals(Numerals.Find(x => x.Numerical.Equals(100)).Word))  {  number.ErrorCode = 23;  number.ErrorWord = split[1];  return number;  }  return number;  }    public Number GetNumberRomanian(Number number)  {  //  var romanNumerals = new string[][]  {  new string[]{"", "I", "II", "III", "IV", "V", "VI", "VII", "VIII", "IX"}, // ones  new string[]{"", "X", "XX", "XXX", "XL", "L", "LX", "LXX", "LXXX", "XC"}, // tens  new string[]{"", "C", "CC", "CCC", "CD", "D", "DC", "DCC", "DCCC", "CM"}, // hundreds  new string[]{"", "M", "MM", "MMM"} // thousands  };  var intArr = number.num.ToString().Reverse().ToArray();  var len = intArr.Length;  var romanNumeral = "";  var i = len;  while (i-- > 0)  {  romanNumeral += romanNumerals[i][Int32.Parse(intArr[i].ToString())];  }  number.romanian = romanNumeral;  return number;  }  }  } |

Задание 2

**Script.js**

|  |
| --- |
| class ChessLine {  constructor() {  this.line1 = "";  this.line2 = "";  }  }  function ShuffleLine() {  let data = new ChessLine();  data.line1 = document.getElementById("input1").value;  data.line2 = document.getElementById("input2").value;  let url = "http://91.240.86.250:9092/api/shuffle";  fetch(url, {  method: 'POST', // \*GET, POST, PUT, DELETE, etc.  mode: 'cors', // no-cors, cors, \*same-origin  headers: {  'Content-Type': 'application/json',  // 'Content-Type': 'application/x-www-form-urlencoded',  },  body: JSON.stringify(data), // body data type must match "Content-Type" header  })  .then(response => response.json()).then(json => {  result = json.result; console.log(result);  document.getElementById("output").innerHTML = result;  console.log(json);  });  } |

**Handler.cs**

|  |
| --- |
| public ChessLine GetShuffledLine(ChessLine chessLine)  {  chessLine.line1 = Regex.Replace(chessLine.line1, @"\s\s+", " ").Trim();  chessLine.line2 = Regex.Replace(chessLine.line2, @"\s\s+", " ").Trim();  List<string> split1 = chessLine.line1.Split(" ").ToList();  List<string> split2 = chessLine.line2.Split(" ").ToList();  List<string> split = new List<string>();  if (split1.Count == 0)  {  split.AddRange(split2);  }  else  {  for (int i = 0, j = 0; i < split1.Count || j < split2.Count; i++, j++)  {  if (i < split1.Count)  split.Add(split1[i]);  if (j < split2.Count)  split.Add(split2[j]);  }  }  for (int i = 0; i < split.Count - 1; i++)  {  split[i] += " ";  }  chessLine.Result = string.Concat(split).TrimEnd();  return chessLine**;**  **}** |