**Министерство образования и науки Российской Федерации**

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ**

Факультет программной инженерии и компьютерной техники

Кафедра информатики и прикладной математики   
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Алгоритмы и структуры данных»

**ОТЧЁТ**

по лабораторной работе №4  
неделя четвертая

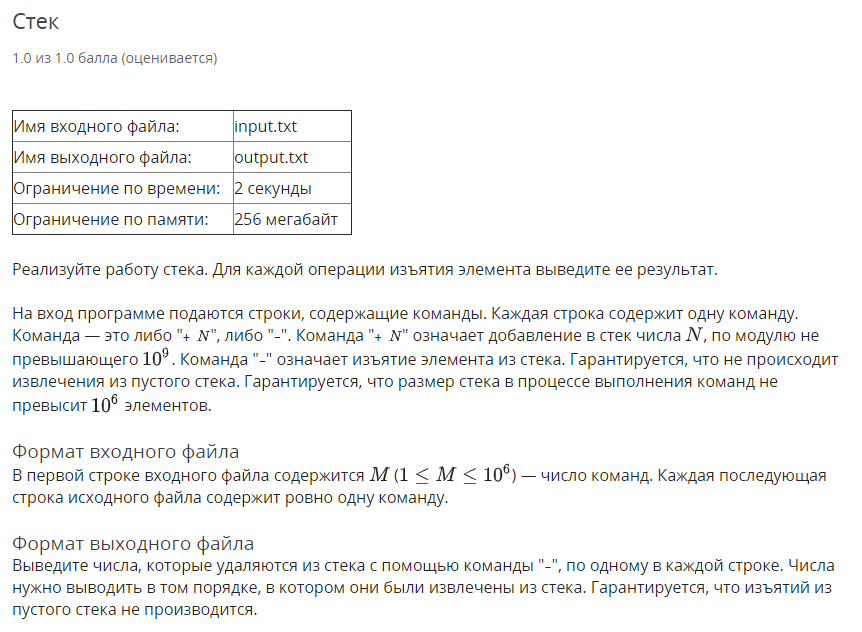
Выполнил:  
Айгузин Иван Олегович   
P3218

Преподаватели:

Романов Алексей Андреевич  
Волчек Дмитрий Геннадьевич

Санкт-Петербург

2018



using System;

using System.IO;

using System.Linq;

namespace Week04.Task01 {

public sealed class Program {

private static StreamReader In;

private static StreamWriter Out;

private static void Main(string[] args) {

if (!args.Contains("console")) {

SetupIO();

}

Run();

if (args.Contains("console")) {

Console.ReadLine();

}

DisposeIO();

}

private static void Run() {

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

var stack = new string[1000000];

var cursor = 0;

for (var i = 0; i < n; i++) {

var line = ReadLineArray();

if (line[0] == "+") {

stack[cursor++] = line[1];

continue;

}

Console.WriteLine(stack[--cursor]);

}

}

private static string[] ReadLineArray() {

return Console.ReadLine()

.Split(' ')

.ToArray();

}

private static void SetupIO() {

In = new StreamReader("input.txt");

Out = new StreamWriter("output.txt");

Console.SetIn(In);

Console.SetOut(Out);

}

private static void DisposeIO() {

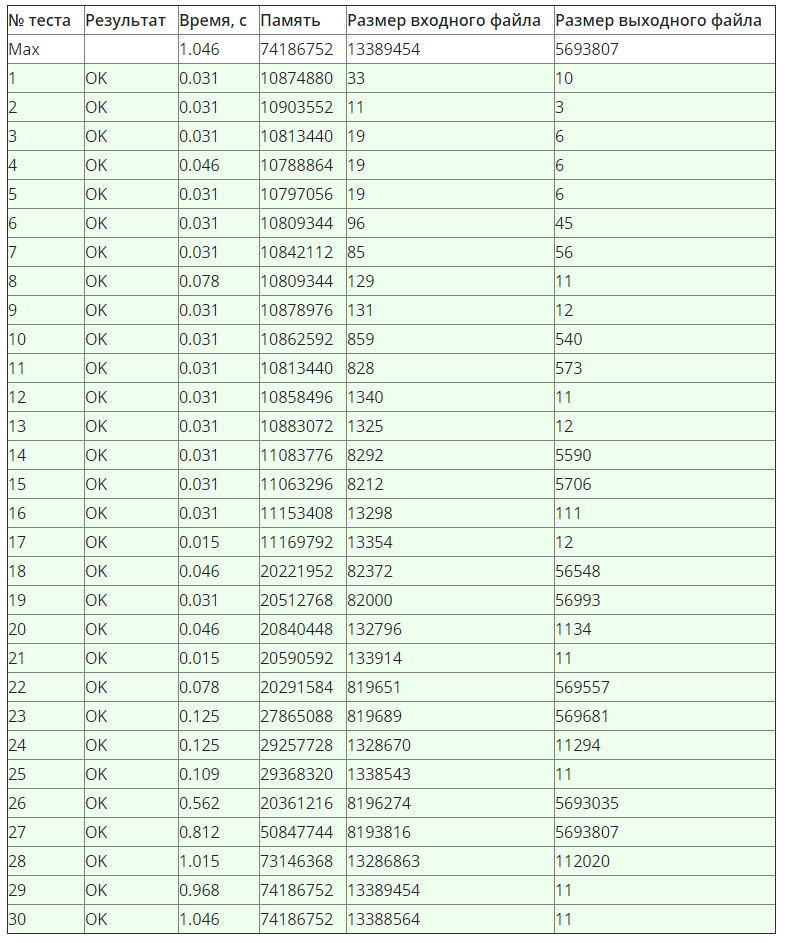
In?.Dispose();

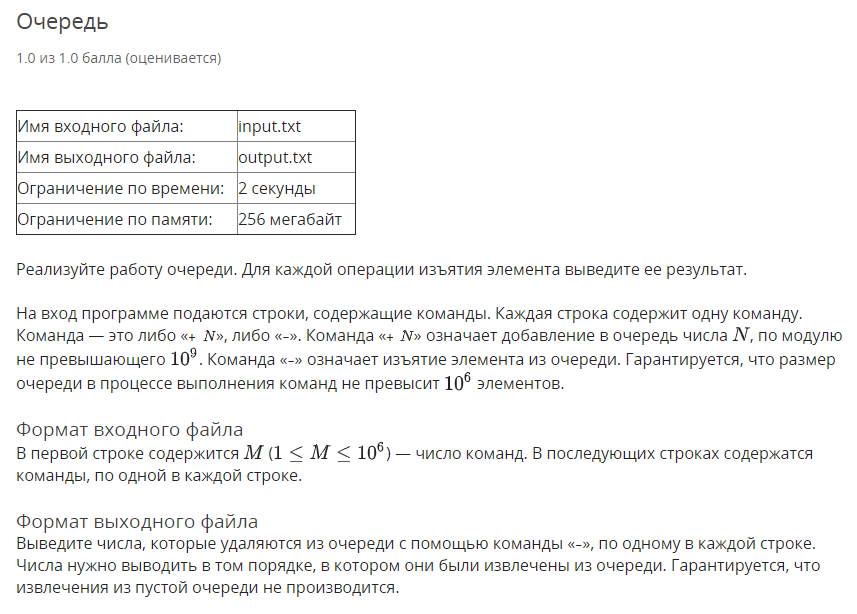
Out?.Dispose();

}

}

}





using System;

using System.IO;

using System.Linq;

namespace Week04.Task02 {

public sealed class Program {

private static StreamReader In;

private static StreamWriter Out;

private static void Main(string[] args) {

if (!args.Contains("console")) {

SetupIO();

}

Run();

if (args.Contains("console")) {

Console.ReadLine();

}

DisposeIO();

}

private static void Run() {

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

var queue = new string[1000000];

var cursorTop = 0;

var cursorBottom = 0;

for (var i = 0; i < n; i++) {

var line = ReadLineArray();

if (line[0] == "+") {

queue[cursorTop++] = line[1];

continue;

}

Console.WriteLine(queue[cursorBottom++]);

}

}

private static string[] ReadLineArray() {

return Console.ReadLine()

.Split(' ')

.ToArray();

}

private static void SetupIO() {

In = new StreamReader("input.txt");

Out = new StreamWriter("output.txt");

Console.SetIn(In);

Console.SetOut(Out);

}

private static void DisposeIO() {

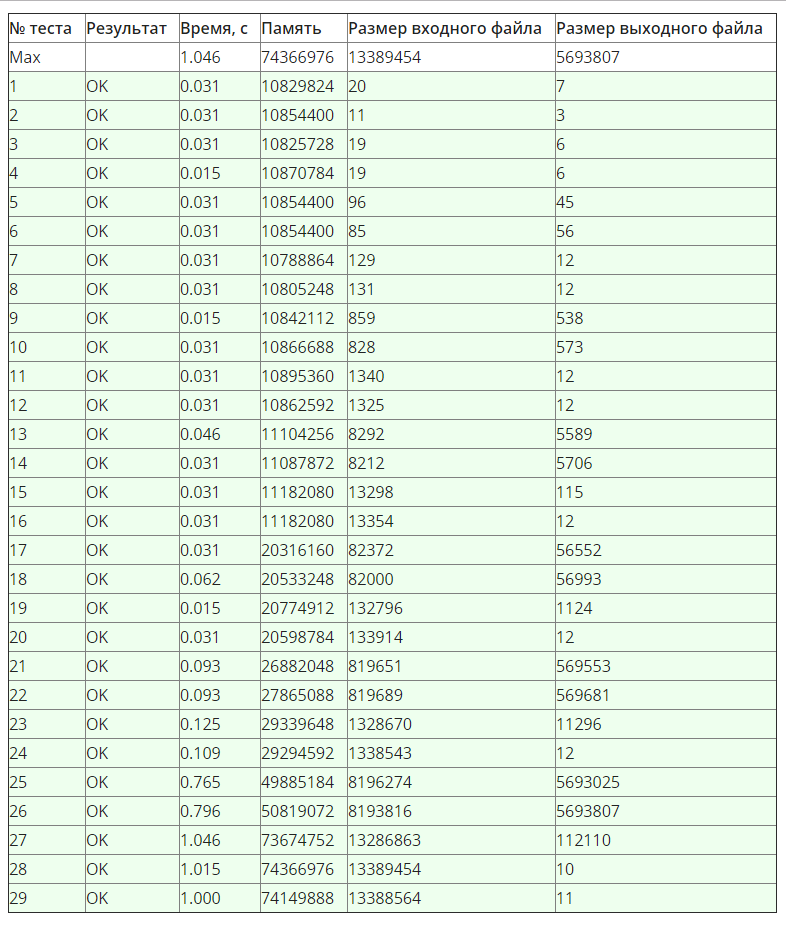
In?.Dispose();

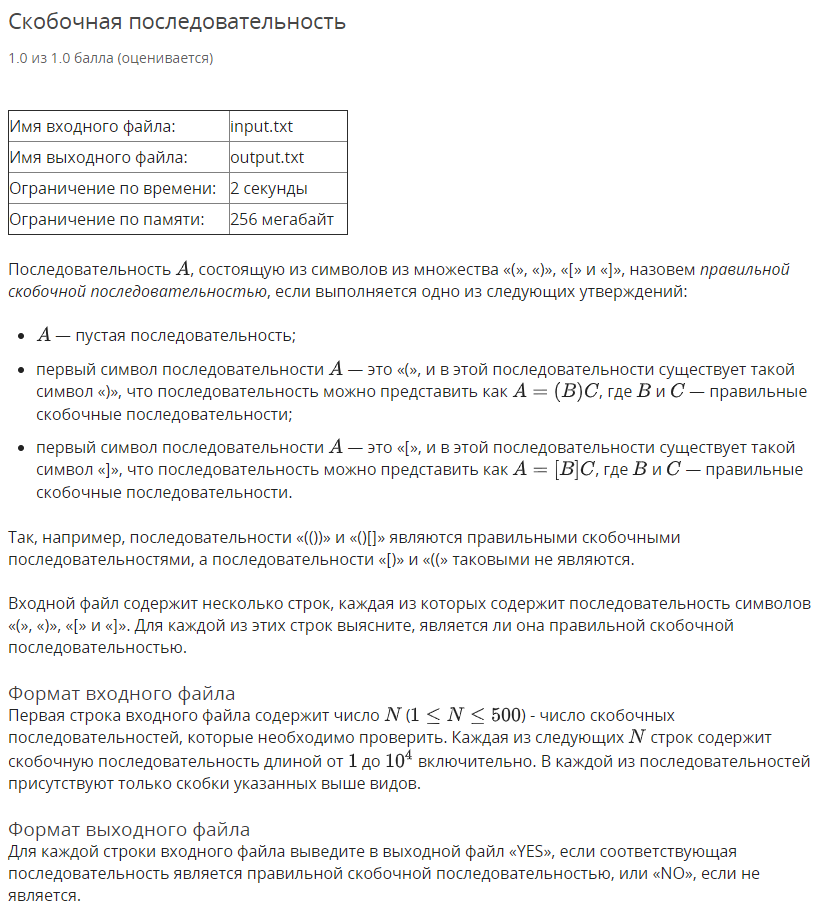
Out?.Dispose();

}

}

}





using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace Week04.Task03 {

public sealed class Program {

private static StreamReader \_in;

private static StreamWriter \_out;

private static void Main(string[] args) {

if (!args.Contains("console")) {

SetupIO();

}

Run();

if (args.Contains("console")) {

Console.ReadLine();

}

DisposeIO();

}

private static void Run() {

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

var stack = new Stack<char>();

var braces = new Dictionary<char, char> {

['('] = ')',

['['] = ']'

};

for (var i = 0; i < n; i++) {

stack.Clear();

CheckLine(Console.ReadLine());

}

void CheckLine(string line) {

foreach (var c in line) {

if (braces.ContainsKey(c)) {

stack.Push(c);

continue;

}

if (stack.Count == 0) {

Console.WriteLine("NO");

return;

}

if (braces[stack.Pop()] == c) {

continue;

}

Console.WriteLine("NO");

return;

}

Console.WriteLine(stack.Count == 0 ? "YES" : "NO");

}

}

private static string[] ReadLineArray() {

return Console.ReadLine()

.Split(' ')

.ToArray();

}

private static void SetupIO() {

\_in = new StreamReader("input.txt");

\_out = new StreamWriter("output.txt");

Console.SetIn(\_in);

Console.SetOut(\_out);

}

private static void DisposeIO() {

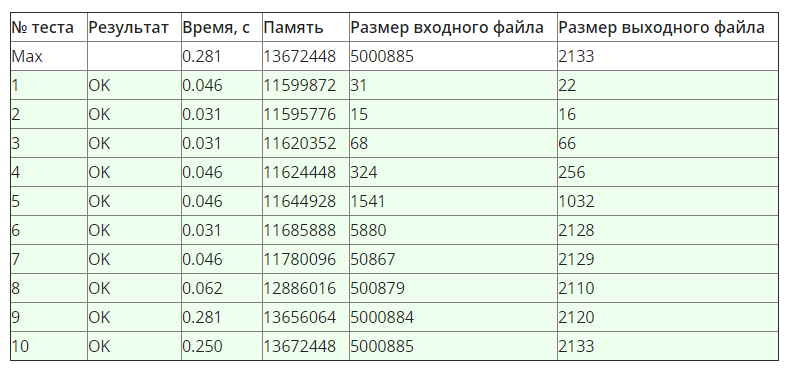
\_in?.Dispose();

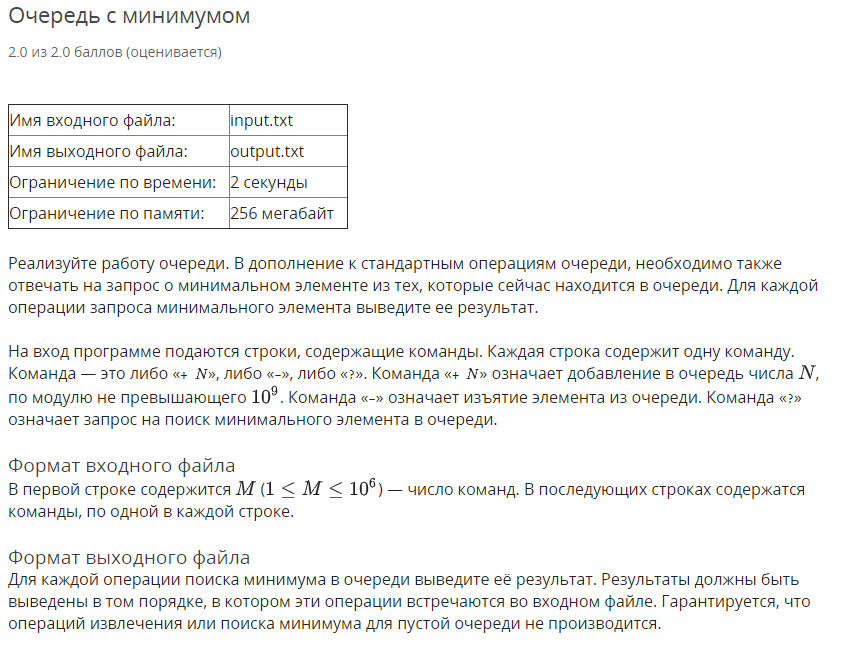
\_out?.Dispose();

}

}

}





using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace Week04.Task04 {

public sealed class Program {

private static StreamReader \_in;

private static StreamWriter \_out;

private static void Main(string[] args) {

if (!args.Contains("console")) {

SetupIO();

}

Run();

if (args.Contains("console")) {

Console.ReadLine();

}

DisposeIO();

}

private static void Run() {

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

var queue = new Queue<int>();

var mins = new LinkedList<int>();

for (var i = 0; i < n; i++) {

var line = ReadLineArray();

switch (line[0]) {

case "+":

var a = int.Parse(line[1]);

queue.Enqueue(a);

while (mins.Count > 0 && mins.First.Value > a) {

mins.RemoveFirst();

}

mins.AddFirst(a);

break;

case "-":

var b = queue.Dequeue();

if (mins.Last.Value == b) {

mins.RemoveLast();

}

break;

default:

Console.WriteLine(mins.Last.Value);

break;

}

}

}

private static string[] ReadLineArray() {

return Console.ReadLine()

.Split(' ')

.ToArray();

}

private static void SetupIO() {

\_in = new StreamReader("input.txt");

\_out = new StreamWriter("output.txt");

Console.SetIn(\_in);

Console.SetOut(\_out);

}

private static void DisposeIO() {

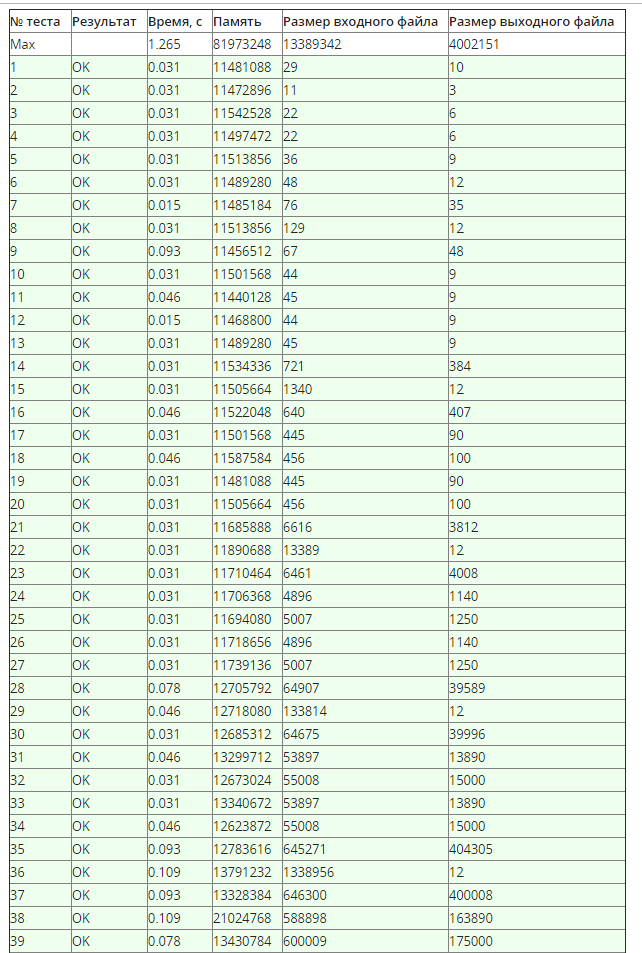
\_in?.Dispose();

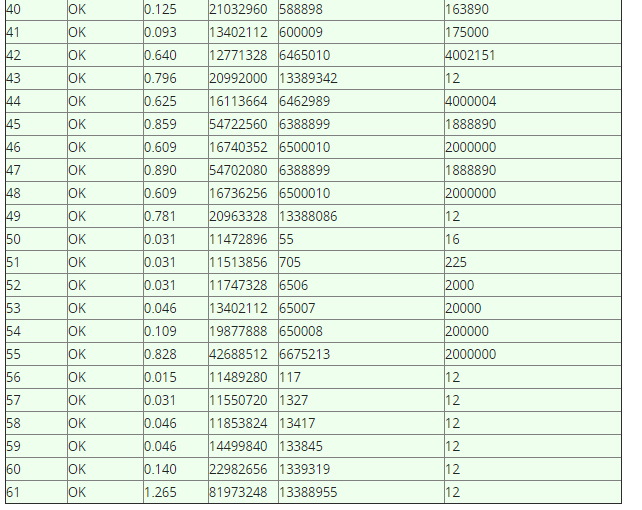
\_out?.Dispose();

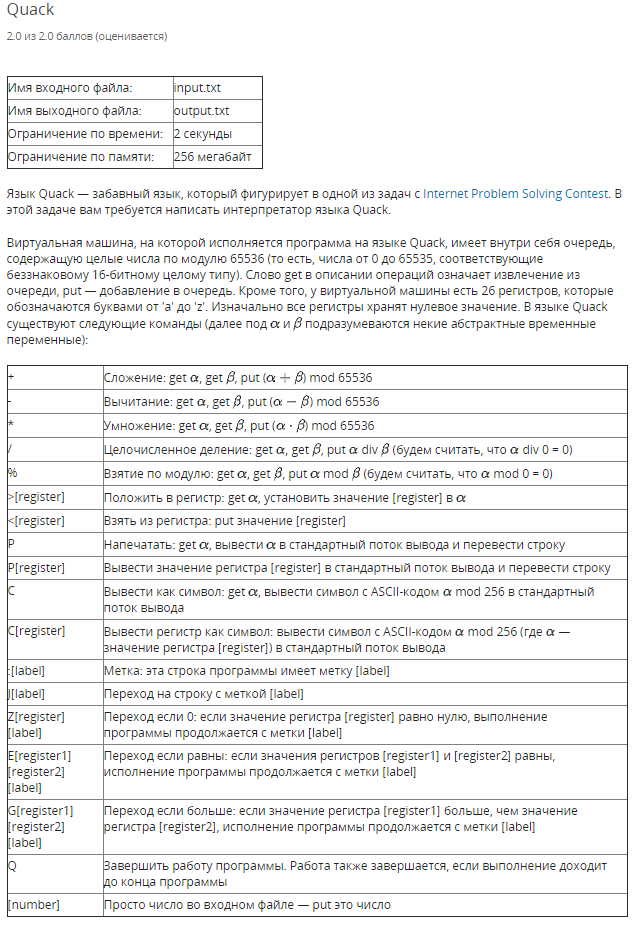
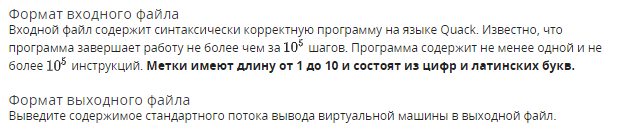
}

}

}





using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace Week04.Task05 {

public sealed class Program {

private static StreamWriter \_out;

private static void Main(string[] args) {

if (!args.Contains("console")) {

SetupIO();

}

new Quack().Run(File.ReadAllLines("input.txt"));

if (args.Contains("console")) {

Console.ReadLine();

}

DisposeIO();

}

private static void SetupIO() {

\_out = new StreamWriter("output.txt");

Console.SetOut(\_out);

}

private static void DisposeIO() {

\_out?.Dispose();

}

}

internal class Quack {

private readonly Queue<ushort> \_queue = new Queue<ushort>();

private Dictionary<int, ushort> \_registers;

private int \_cursor;

private Dictionary<char, Action<string>> \_instructions;

private Dictionary<string, int> \_labels;

private bool \_stopped;

public void Run(string[] lines) {

\_registers = Enumerable.Range('a', 26)

.ToDictionary(x => x, y => (ushort) 0);

\_queue.Clear();

DefineLabels(lines);

DefineInstructions();

while (\_cursor < lines.Length && !\_stopped) {

Interpret(lines[\_cursor]);

\_cursor += 1;

}

}

private void Interpret(string s) {

if (char.IsDigit(s[0])) {

\_queue.Enqueue(ushort.Parse(s));

return;

}

\_instructions[s[0]](s);

}

private void DefineInstructions() {

\_instructions = new Dictionary<char, Action<string>> {

['+'] = s => { Put(Get() + Get()); },

['-'] = s => { Put(Get() - Get()); },

['\*'] = s => { Put(Get() \* Get()); },

['/'] = s => {

var a = Get();

var b = Get();

Put(b == 0 ? 0 : a / b);

},

['%'] = s => {

var a = Get();

var b = Get();

Put(b == 0 ? 0 : a % b);

},

['>'] = s => { \_registers[s[1]] = Get(); },

['<'] = s => { Put(\_registers[s[1]]); },

['P'] = s => { Console.WriteLine(s.Length == 1 ? Get() : \_registers[s[1]]); },

['C'] = s => { Console.Write((char) ((s.Length == 1 ? Get() : \_registers[s[1]]) % 256)); },

[':'] = s => { },

['J'] = s => { \_cursor = \_labels[new string(s.Skip(1).ToArray())]; },

['Z'] = s => {

if (\_registers[s[1]] == 0) {

\_cursor = \_labels[new string(s.Skip(2).ToArray())];

}

},

['E'] = s => {

if (\_registers[s[1]] == \_registers[s[2]]) {

\_cursor = \_labels[new string(s.Skip(3).ToArray())];

}

},

['G'] = s => {

if (\_registers[s[1]] > \_registers[s[2]]) {

\_cursor = \_labels[new string(s.Skip(3).ToArray())];

}

},

['Q'] = s => { \_stopped = true; }

};

ushort Get() {

return \_queue.Dequeue();

}

void Put(int value) {

\_queue.Enqueue((ushort) (value % 65536));

}

}

private void DefineLabels(string[] lines) {

\_labels = lines

.Select((s, i) => (label: s, line: i))

.Where(s => s.label[0] == ':')

.ToDictionary(tuple => new string(tuple.label.Skip(1).ToArray()), tuple => tuple.line);

}

}

}

