Министерство образования и науки Российской Федерации

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

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «**Алгоритмы и структуры данных**»

ОТЧЁТ

неделя №4

Студент: **Сараев В.В. группы Р3218**

Преподаватели:

Романов Алексей Андреевич

Волчек Дмитрий Геннадьевич

Санкт-Петербург

2019 г.

Задача 1:

Реализуйте работу стека. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо "+ N", либо "-". Команда "+ N" означает добавление в стек числа N, по модулю не превышающего 10^9. Команда "-" означает изъятие элемента из стека. Гарантируется, что не происходит извлечения из пустого стека. Гарантируется, что размер стека в процессе выполнения команд не превысит 10^6 элементов.

Формат входного файла

В первой строке входного файла содержится M ($1 \leq M \leq 10^6$) — число команд. Каждая последующая строка исходного файла содержит ровно одну команду.

Формат выходного файла

Выведите числа, которые удаляются из стека с помощью команды "-", по одному в каждой строке. Числа нужно выводить в том порядке, в котором они были извлечены из стека. Гарантируется, что изъятий из пустого стека не производится.

Решение:

```
import java.io.*;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;


public class Main {


    public static void main(String[] args) throws IOException {
        String content;
        int counter = 0;
                // StringBuilder для сокращения времени конкатенации строк
        StringBuilder deleted = new StringBuilder();
                //открытие файла для чтения
        BufferedReader br = new BufferedReader(new FileReader("input.txt"));


        int[] stack = new int[Integer.valueOf(br.readLine())];
                //последовательное считывание и обработка данных
        while ((content = br.readLine()) != null) {
                        //удаление элемента из начала стека исходя из команды
            if(content.split(" ")[0].equals("-")){
                counter--;
                deleted.append(stack[counter]).append("\n");
            }
                        //добавление элементав начало стека исходя из команды
```

```
            if(content.split(" ")[0].equals("+")){
                stack[counter] =  Integer.valueOf(content.split(" ")[1].trim());
                counter++;
            }
        }
                // запись в файл
        Path outputFilePath = Paths.get("output.txt");
        Files.write(outputFilePath, deleted.toString().getBytes());
    }
}
```

Результат:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.375	204255232	13389454	5193807
1	OK	0.125	22007808	33	8
2	OK	0.125	21950464	11	2
3	OK	0.109	22007808	19	4
4	OK	0.140	22056960	19	4
5	OK	0.187	22020096	19	4
6	OK	0.171	22081536	96	41
7	OK	0.125	22020096	85	51
8	OK	0.140	22065152	129	10
9	OK	0.203	22061056	131	11
10	OK	0.140	22089728	859	493
11	OK	0.140	22122496	828	523
12	OK	0.156	22175744	1340	10
13	OK	0.187	22134784	1325	11
14	OK	0.140	24023040	8292	5099
15	OK	0.156	23990272	8212	5206
16	OK	0.187	24350720	13298	101
17	OK	0.156	24371200	13354	11
18	OK	0.281	33333248	82372	51585
19	OK	0.250	32083968	82000	51993
20	OK	0.265	36646912	132796	1035
21	OK	0.296	36438016	133914	10
22	OK	0.593	59494400	819651	519561
23	OK	0.562	59568128	819689	519681
24	OK	0.515	60350464	1328670	10303
25	OK	0.578	60252160	1338543	10
26	OK	1.109	184713216	8196274	5193183
27	OK	1.218	174391296	8193816	5193807
28	OK	1.375	204255232	13286863	102183
29	OK	1.234	203755520	13389454	10
30	OK	1.203	203784192	13388564	10

Задание 2:

Реализуйте работу очереди. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо «+ N», либо «–». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего 10^9. Команда «–» означает изъятие элемента из очереди. Гарантируется, что размер очереди в процессе выполнения команд не превысит 10^6 элементов.

Формат входного файла

В первой строке содержится M ($1 \leq M \leq 10^6$) — число команд. В последующих строках содержатся команды, по одной в каждой строке.

Формат выходного файла

Выведите числа, которые удаляются из очереди с помощью команды «–», по одному в каждой строке. Числа нужно выводить в том порядке, в котором они были извлечены из очереди. Гарантируется, что извлечения из пустой очереди не производится.

Решение:

```
import java.io.*;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Collections;
public class Main {
    public static void main(String[] args) throws IOException {
        String content;
                //указатель хвоста очереди
        int tail = 0;
                //указатель головы очереди
        int head = 0;
        StringBuilder deleted = new StringBuilder();
        BufferedReader br = new BufferedReader(new FileReader("input.txt"));
        int len = Integer.valueOf(br.readLine());
        int[] queue = new int[len];
                //последовательное чтение и обработка данных
        while ((content = br.readLine()) != null) {
            if(content.split(" ")[0].equals("+")){
        //добавяем элемент в хвост очереди (кольцевого буфера)
        //если позиция хвоста равна размеру очереди, то перекидывем указатель хвоста на начало очереди
                if (++tail == len)
                    tail = 0;
                queue[tail] = Integer.valueOf(content.split(" ")[1]);
            }
            else{
```

```
                //удаляем элемент из головы очереди
                /если позиция головы равна размеру очереди, то также перекидывем на начало
                if (++head == len) {
                    head = 0;
                }
                deleted.append(queue[head]).append("\n");
            }
        }
                //запись в файл
        Path outputFilePath = Paths.get("output.txt");
        Files.write(outputFilePath, Collections.singleton(deleted));
    }
}
```

Результат:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.031	183566336	13389454	5193809
1	OK	0.125	22102016	20	7
2	OK	0.125	21995520	11	4
3	OK	0.125	22065152	19	6
4	OK	0.140	22077440	19	6
5	OK	0.187	22056960	96	43
6	OK	0.156	22089728	85	53
7	OK	0.156	22036480	129	13
8	OK	0.125	22044672	131	13
9	OK	0.109	22155264	859	493
10	OK	0.140	22110208	828	525
11	OK	0.125	22171648	1340	13
12	OK	0.156	22130688	1325	13
13	OK	0.140	23859200	8292	5100
14	OK	0.156	23605248	8212	5208
15	OK	0.156	24244224	13298	107
16	OK	0.140	24244224	13354	13
17	OK	0.218	31490048	82372	51591
18	OK	0.250	30707712	82000	51995

19	OK	0.234	32731136	132796	1027
20	OK	0.218	32522240	133914	13
21	OK	0.500	60280832	819651	519559
22	OK	0.500	60772352	819689	519683
23	OK	0.484	58843136	1328670	10307
24	OK	0.453	58478592	1338543	13
25	OK	1.015	183566336	8196274	5193175
26	OK	0.984	170057728	8193816	5193809
27	OK	1.031	169496576	13286863	102275
28	OK	1.015	169381888	13389454	11
29	OK	1.000	169730048	13388564	12

Задание 3:

Последовательность A, состоящую из символов из множества «(», «)», «[» и «]», назовем *правильной скобочной последовательностью*, если выполняется одно из следующих утверждений:

- A — пустая последовательность;
- первый символ последовательности A — это «(», и в этой последовательности существует такой символ «)», что последовательность можно представить как $A = (B)C$, где B и C — правильные скобочные последовательности;
- первый символ последовательности A — это «[», и в этой последовательности существует такой символ «]», что последовательность можно представить как $A = [B]C$, где B и C — правильные скобочные последовательности.

Так, например, последовательности «(())» и «()[]» являются правильными скобочными последовательностями, а последовательности «[)» и «((» таковыми не являются.

Входной файл содержит несколько строк, каждая из которых содержит последовательность символов «(», «)», «[» и «]». Для каждой из этих строк выясните, является ли она правильной скобочной последовательностью.

Формат входного файла

Первая строка входного файла содержит число N ($1 \le N \le 500$) - число скобочных последовательностей, которые необходимо проверить. Каждая из следующих N строк содержит скобочную последовательность длиной от 1 до 10^4 включительно. В каждой из последовательностей присутствуют только скобки указанных выше видов.

Формат выходного файла

Для каждой строки входного файла выведите в выходной файл «YES», если соответствующая последовательность является правильной скобочной последовательностью, или «NO», если не является.

Решение:

```
import java.io.*;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Collections;
public class Main {
    public static void main(String[] args) throws IOException {
        String content;
        StringBuilder info = new StringBuilder();
        BufferedReader br = new BufferedReader(new FileReader("input.txt"));
        br.readLine();
        while ((content = br.readLine()) != null) {
            if(isSeqTrue(content)){
                info.append("YES").append("\n");
            }
            else {info.append("NO").append("\n"); }
        }
        Path outputFilePath = Paths.get("output.txt");
        Files.write(outputFilePath, Collections.singleton(info));
    }
    private static boolean isSeqTrue(String content) {
        //массив служащий для описания структуры скобочной последовательности
        int[] bracketsStruct = new int[content.length()];
        int marck = 0;
        //если пос-ть сразу начинается с закрывающий скобок,
        //то она сразу считается неккоректной
        if (content.charAt(0) == ')' || content.charAt(0) == ']') {
            return false;
        }
        //определение типа скобок
        for (int i = 0; i < content.length(); i++) {
                //если исследуемый символ совпададет с одним из типов открывающих скобок, то
                //заносим 1 в bracketsStruct как символ описывающий тип скобки и
                // увеличиваем marck для нахождения кол-ва открывающих скобок стоящих друг за другом
            if (content.charAt(i) == '(') {
                bracketsStruct[marck] = 1;
                marck++;
            }
            if (content.charAt(i) == '[') {
```

```
                bracketsStruct[marck] = 0;
                marck++;
            }
                //проверяем на последовательность закрывающих скобок
                //если marck == 0 или bracketsStruct[marck – 1] == 0,
                //то открывающих скобок не найдено, иначе уменьшаем marck
                //для проверки равновесия структуры последовательности
            if (content.charAt(i) == ')') {
                if (marck == 0 || bracketsStruct[marck - 1] == 0) {
                    return false;
                }
                else marck--;
            }
            if (content.charAt(i) == ']') {
                if (marck == 0 || bracketsStruct[marck - 1] == 1){
                    return false;
                }
                else marck--;
            }
        }
        //marck == 0 то, кол-во закрывающих и открывающих скобок равно =>
        //последовательность корректна
        return marck == 0;
    }
}
```

Результат:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.312	56516608	5000885	1635
1	OK	0.140	22097920	31	19
2	OK	0.125	22069248	15	14
3	OK	0.125	22102016	68	52
4	OK	0.125	22052864	324	194
5	OK	0.125	22065152	1541	778
6	OK	0.140	22274048	5880	1630
7	OK	0.140	23977984	50867	1631
8	OK	0.218	27959296	500879	1612
9	OK	0.312	56516608	5000884	1622
10	OK	0.296	56348672	5000885	1635

Задание 4:

Реализуйте работу очереди. В дополнение к стандартным операциям очереди, необходимо также отвечать на запрос о минимальном элементе из тех, которые сейчас находится в очереди. Для каждой операции запроса минимального элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо «+ N», либо «–», либо «?». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего 10^9. Команда «–» означает изъятие элемента из очереди. Команда «?» означает запрос на поиск минимального элемента в очереди.

Формат входного файла

В первой строке содержится M ($1 \leq M \leq 10^6$) — число команд. В последующих строках содержатся команды, по одной в каждой строке.

Формат выходного файла

Для каждой операции поиска минимума в очереди выведите её результат. Результаты должны быть выведены в том порядке, в котором эти операции встречаются во входном файле. Гарантируется, что операций извлечения или поиска минимума для пустой очереди не производится.

Решение:

```
#include "edx-io.hpp"

#include "stdafx.h"

#include <iostream>

#include <conio.h>

#include <cstdlib>

#include <fstream>

#include <algorithm>

using namespace std;


int main() {


        string sign;

        //указатели на хвост и голову очереди

        int tail = 0;

        int     head = 0;

        //счетчики добавления и удаления

        int     counterPlus = 0;

        int     counterMinus = 0;

        int     len = 0;

        int     value = 0;

        io >> len;

        int* queue = new int[len];

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

                io >> sign;

                if (sign == "+") {
```

```
                        if (++tail == len)
                                tail = 0;
                        io >> value;
                        queue[tail] = value;
                        counterPlus++;
                }
                if (sign == "-") {
                        if (++head == len) {
                                head = 0;
                        }
                        counterMinus++;
                }
                if (sign == "?") {
                        //находим длину дейсвительной очереди
                        int tmplen = (counterPlus + 1) - (counterMinus + 1);
                        int *tmpArray = new int[tmplen];
                        int tmp = counterMinus + 1;
                        //копируем дейсвительную последовательность во временный массив
                        for (int i = 0; i < counterPlus+1; i++) {
                                tmpArray[i] = queue[tmp++];
                        }
                        //сортируем массив и выводим минмальный элемент массива,
                        //sa по совместительству и очереди
                        sort(tmpArray, tmpArray + tmplen);
                        io << tmpArray[0] << "\n";
                }
        }
        return 0;
}
```

Результат:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.125	23203840	13389342	4002151
1	OK	0.000	2224128	29	10
2	OK	0.015	2220032	11	3
3	OK	0.015	2224128	22	6
4	OK	0.015	2224128	22	6
5	OK	0.000	2224128	36	9

10	OK	0.000	2224128	44	9
11	OK	0.000	2220032	45	9
12	OK	0.000	2220032	44	9
13	OK	0.000	2220032	45	9
14	OK	0.015	2236416	721	384
15	OK	0.015	2236416	1340	12
16	OK	0.000	2224128	640	407
17	OK	0.015	2240512	445	90
18	OK	0.000	2224128	456	100
19	OK	0.000	2224128	445	90
20	OK	0.015	2224128	456	100
21	OK	0.000	2244608	6616	3812
22	OK	0.000	2236416	13389	12
23	OK	0.015	2236416	6461	4008
24	OK	0.000	2236416	4896	1140
25	OK	0.015	2236416	5007	1250
26	OK	0.000	2236416	4896	1140
27	OK	0.000	2248704	5007	1250
28	OK	0.000	2269184	64907	39589
29	OK	0.000	2277376	133814	12
30	OK	0.000	2252800	64675	39996

31	OK	0.015	2285568	53897	13890
32	OK	0.000	2269184	55008	15000
33	OK	0.000	2273280	53897	13890
34	OK	0.000	2273280	55008	15000
35	OK	0.015	2613248	645271	404305
36	OK	0.015	3555328	1338956	12
37	OK	0.015	2609152	646300	400008
38	OK	0.000	2813952	588898	163890
39	OK	0.015	2633728	600009	175000
40	OK	0.000	2813952	588898	163890

41	OK	0.015	2629632	600009	175000
42	OK	0.125	9621504	6465010	4002151
43	OK	0.062	19206144	13389342	12
44	OK	0.109	9617408	6462989	4000004
45	OK	0.078	12206080	6388899	1888890
46	OK	0.078	10321920	6500010	2000000
47	OK	0.078	12210176	6388899	1888890
48	OK	0.078	10321920	6500010	2000000
49	OK	0.062	19206144	13388086	12
50	OK	0.000	2224128	55	16
51	OK	0.015	2236416	705	225
52	OK	0.000	2248704	6506	2000
53	OK	0.000	2269184	65007	20000
54	OK	0.015	2879488	650008	200000
55	OK	0.078	12496896	6675213	2000000
56	OK	0.015	2220032	117	12
57	OK	0.015	2224128	1327	12
58	OK	0.000	2232320	13417	12
59	OK	0.000	2306048	133845	12
60	OK	0.015	3952640	1339319	12
61	OK	0.062	23203840	13388955	12

Задание 5:

Язык Quack — забавный язык, который фигурирует в одной из задач с Internet Problem Solving Contest. В этой задаче вам требуется написать интерпретатор языка Quack.

Виртуальная машина, на которой исполняется программа на языке Quack, имеет внутри себя очередь, содержащую целые числа по модулю 65536 (то есть, числа от 0 до 65535, соответствующие беззнаковому 16-битному целому типу). Слово get в описании операций означает извлечение из очереди, put — добавление в очередь. Кроме того, у виртуальной машины есть 26 регистров, которые обозначаются буквами от 'a' до 'z'. Изначально все регистры хранят нулевое значение. В языке Quack существуют следующие команды (далее под α и β подразумеваются некие абстрактные временные переменные):

+	Сложение: get α, get β, put $(\alpha + \beta)$ mod 65536
-	Вычитание: get α, get β, put $(\alpha - \beta)$ mod 65536
*	Умножение: get α, get β, put $(\alpha \cdot \beta)$ mod 65536
/	Целочисленное деление: get α, get β, put α div β (будем считать, что α div 0 = 0)
%	Взятие по модулю: get α, get β, put α mod β (будем считать, что α mod 0 = 0)

>[register]	Положить в регистр: get α, установить значение [register] в α
<[register]	Взять из регистра: put значение [register]
P	Напечатать: get α, вывести α в стандартный поток вывода и перевести строку
P[register]	Вывести значение регистра [register] в стандартный поток вывода и перевести строку
C	Вывести как символ: get α, вывести символ с ASCII-кодом α mod 256 в стандартный поток вывода
C[register]	Вывести регистр как символ: вывести символ с ASCII-кодом α mod 256 (где α — значение регистра [register]) в стандартный поток вывода
:[label]	Метка: эта строка программы имеет метку [label]
J[label]	Переход на строку с меткой [label]
Z[register][label]	Переход если 0: если значение регистра [register] равно нулю, выполнение программы продолжается с метки [label]
E[register1] [register2][label]	Переход если равны: если значения регистров [register1] и [register2] равны, исполнение программы продолжается с метки [label]
G[register1] [register2][label]	Переход если больше: если значение регистра [register1] больше, чем значение регистра [register2], исполнение программы продолжается с метки [label]
Q	Завершить работу программы. Работа также завершается, если выполнение доходит до конца программы
[number]	Просто число во входном файле — put это число

Формат входного файла

Входной файл содержит синтаксически корректную программу на языке Quack. Известно, что программа завершает работу не более чем за 10^5 шагов. Программа содержит не менее одной и не более 10^5 инструкций. **Метки имеют длину от 1 до 10 и состоят из цифр и латинских букв.**

Формат выходного файла

Выведите содержимое стандартного потока вывода виртуальной машины в выходной файл.

Решение:

```
using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace Week4

{

    class Lab4_5

    {

        public static void Main(string[] args)

        {

            //инициализируем поток для записи в файл,задаём кодировку и

            //присваиваем свойсто для стандартного потока
```

```
        StreamWriter sw = new StreamWriter("output.txt");
        Console.OutputEncoding = Encoding.ASCII;
        Console.SetOut(sw);

        //ссчитываем данные и запускаем Quack
        string[] stdin = File.ReadAllLines("input.txt");
        Quack quack = new Quack(stdin);
        quack.Run();
        sw.Dispose();
    }
}
class Quack
{
    //position служит для указания положения в коде Quack
    private static ushort[] registers = new ushort[26];
    private static int position = 0;
    private static Dictionary<string, int> labels = new Dictionary<string, int>();
    private static string[] code;
    private static Queue<ushort> queue = new Queue<ushort>();

    public Quack(string[] input)
    {
        code = input;
        RegisterLabels();
    }

    public void Run()
    {
        for (position = 0; position < code.Length; position++)
        {
            //для каждого состояния вызывается определенный метод
            switch (code[position][0])
            {
                case '+':
                    //кладем на стек А + Б, которые тоже достали из стека (далее по аналогии)
                    queue.Enqueue((ushort)((queue.Dequeue() + queue.Dequeue()) % 65536));
                    break;
                    //min
                case '-':
                    queue.Enqueue((ushort)((queue.Dequeue() - queue.Dequeue()) % 65536));
```

```
    break;
    //mul
case '*':
    queue.Enqueue((ushort)(queue.Dequeue() * queue.Dequeue()));
    break;
    //div
case '/':
    queue.Enqueue((ushort)(queue.Dequeue() / queue.Dequeue()));
    break;
    //mod
case '%':
    queue.Enqueue((ushort)(queue.Dequeue() % queue.Dequeue()));
    break;
    //кладем A в регистр
case '>':
    GetRegister(code[position][1]);
    break;
    //забираем A из регистра
case '<':
    SetRegister(code[position][1]);
    break;
//печатаем A
case 'P':
    //в stdo если P без параметра
    if (code[position].Length == 1)
        PrintValueFromStack();
    else
        //если есть параметр то заносим A в регистр
        PrintValueFromRegister(code[position][1]);
    break;
    //вывод A ASCII кодом
case 'C':
    //в stdo если P без параметра
    if (code[position].Length == 1)
        PrintCharFromStack();
    //если есть параметр то заносим A в регистр
    else
        PrintCharFromRegister(code[position][1]);
    break;
case ':': break;
```

```
            case 'J':
                //jump
                GoTo(position);
                break;
                // per = 0, то переход на метку
            case 'Z':
                GoToIfZeroEqual(position);
                break;
                //per == per2, то переход на метку
            case 'E':
                GoToIfEquals(position);
                break;
                //per > per2, то переход на метку
            case 'G':
                GoToIfMoreThan(position);
                break;
                //exit
            case 'Q':
                Exit();
                break;
                //put
            default: queue.Enqueue(ushort.Parse(code[position])); break;
        }
    }
}
//находим все метки и сохраняем в список
private void RegisterLabels()
{
    for (int i = 0; i < code.Length; i++)
        if (code[i][0] == ':')
            labels.Add(code[i].Remove(0, 1), i);
}
private void GetRegister(char register)
{
    registers[register- 'a'] = queue.Dequeue();
}
private void SetRegister(char register)
{
    queue.Enqueue(registers[register - 'a']);
}
```

```
private void PrintValueFromStack()
{
    ushort a = queue.Dequeue();
    Console.WriteLine(a);
}
private void PrintValueFromRegister(char register)
{
    Console.WriteLine(registers[register - 'a']);
}
private void PrintCharFromStack()
{
    ushort a = queue.Dequeue();
    Console.Write((char)(a % 256));
}
private void PrintCharFromRegister(char register)
{
    Console.Write((char)(registers[register - 'a'] % 256));
}
private void GoTo(int index)
{
    position = labels[code[index].Remove(0, 1)];
}
private void GoToIfZeroEqual(int index)
{
    if (registers[code[index][1]  - 'a'] == 0)
        position = labels[code[index].Remove(0, 2)];
}
private void GoToIfEquals(int index)
{
    if (registers[code[index][1] - 'a'] == registers[code[index][2] - 'a'])
        position = labels[code[index].Remove(0, 3)];
}
private void GoToIfMoreThan(int index)
{
    if (registers[code[index][1]  - 'a'] > registers[code[index][2] - 'a'])
        position = labels[code[index].Remove(0, 3)];
}
private void Exit()
{
    position = int.MaxValue;
```

```
        }
    }
}
```

Результат:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.078	24526848	1349803	250850
1	OK	0.062	11264000	69	6
2	OK	0.015	11202560	232	218
3	OK	0.031	11091968	3	0
4	OK	0.031	11173888	100	19
5	OK	0.031	11980800	56	58890
6	OK	0.015	11628544	67	30000
7	OK	0.031	11595776	67	30000
8	OK	0.031	11624448	55	30000
9	OK	0.031	11231232	461	60
10	OK	0.015	11386880	11235	21
11	OK	0.031	11599872	23748	42
12	OK	0.031	12165120	66906	8905
13	OK	0.046	11071488	7332	954
14	OK	0.046	11046912	4611	602
15	OK	0.031	11620352	37968	5424
16	OK	0.031	10952704	14	2
17	OK	0.031	10940416	70	10
18	OK	0.046	10956800	350	50
19	OK	0.031	10981376	1750	250
20	OK	0.031	11108352	8750	1250
21	OK	0.031	11730944	43750	6250
22	OK	0.031	14876672	218750	31250
23	OK	0.031	11534336	34606	4721
24	OK	0.062	18481152	683180	7
25	OK	0.062	18448384	683102	0
26	OK	0.078	24526848	1349803	0
27	OK	0.062	19107840	491572	247791
28	OK	0.062	19165184	491488	249618
29	OK	0.062	19083264	491600	249600
30	OK	0.078	19099648	491502	250850
31	OK	0.062	19099648	491416	249477
32	OK	0.078	19128320	491520	250262
33	OK	0.078	19124224	491317	246859
34	OK	0.062	19099648	491514	248199
35	OK	0.062	19103744	491557	249601