**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ**

**Факультет программной инженерии и компьютерной техники**

# ЛАБОРАТОРНАЯ РАБОТА № 4

**ПО ДИСЦИПЛИНЕ «Алгоритмы и структуры данных»**

Выполнил: Мещеряков Владимир Алевтинович

Группа: P3218

Преподаватель: Романов Алексей Андреевич

Волчек Дмитрий Геннадьевич

Санкт-Петербург

2019 г.

# Week 4

## 1) Стек

| | |
|---|---|
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

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 ($1 \le M \le 10^6$) — 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.

### Пример

| input.txt | output.txt |
|---|---|
| 6<br>+ 1<br>+ 10<br>-<br>+ 2<br>+ 1234<br>- | 10<br>1234 |

```
#include "edx-io.hpp"
using namespace std;

int main() {

	long N;
	io >> N;
	char action;

	long* stack = new long[N];
	long top = -1;

	for (long i = 0; i < N; i++) {
		io >> action;
		if (action == '+') {
			top++;
			io >> stack[top];
		}
		else {
			io << stack[top] << '\n';
			top--;
		}
	}
	return 0;
}
```

| № теста | Результат | Время, с | Память | Размер входного файла | Размер выходного файла |
|---|---|---|---|---|---|
| Max | | 0.156 | 19202048 | 13389454 | 5693807 |
| 1 | OK | 0.000 | 2449408 | 33 | 10 |
| 2 | OK | 0.015 | 2449408 | 11 | 3 |
| 3 | OK | 0.015 | 2445312 | 19 | 6 |
| 4 | OK | 0.015 | 2449408 | 19 | 6 |
| 5 | OK | 0.000 | 2449408 | 19 | 6 |
| 6 | OK | 0.000 | 2445312 | 96 | 45 |
| 7 | OK | 0.015 | 2461696 | 85 | 56 |
| 8 | OK | 0.015 | 2453504 | 129 | 11 |
| 9 | OK | 0.015 | 2461696 | 131 | 12 |
| 10 | OK | 0.015 | 2461696 | 859 | 540 |
| 11 | OK | 0.000 | 2449408 | 828 | 573 |
| 12 | OK | 0.000 | 2445312 | 1340 | 11 |
| 13 | OK | 0.000 | 2228224 | 1325 | 12 |
| 14 | OK | 0.000 | 2240512 | 8292 | 5590 |
| 15 | OK | 0.000 | 2228224 | 8212 | 5706 |
| 16 | OK | 0.000 | 2228224 | 13298 | 111 |
| 17 | OK | 0.000 | 2228224 | 13354 | 12 |
| 18 | OK | 0.000 | 2228224 | 82372 | 56548 |
| 19 | OK | 0.000 | 2244608 | 82000 | 56993 |
| 20 | OK | 0.000 | 2281472 | 132796 | 1134 |
| 21 | OK | 0.000 | 2265088 | 133914 | 11 |
| 22 | OK | 0.015 | 2637824 | 819651 | 569557 |
| 23 | OK | 0.015 | 2834432 | 819689 | 569681 |
| 24 | OK | 0.015 | 3538944 | 1328670 | 11294 |
| 25 | OK | 0.015 | 3551232 | 1338543 | 11 |
| 26 | OK | 0.140 | 10014720 | 8196274 | 5693035 |
| 27 | OK | 0.156 | 12013568 | 8193816 | 5693807 |
| 28 | OK | 0.062 | 19025920 | 13286863 | 112020 |
| 29 | OK | 0.062 | 19202048 | 13389454 | 11 |
| 30 | OK | 0.062 | 19202048 | 13388564 | 11 |

# 2)Очередь

1.0 из 1.0 балла (оценивается)

| | |
|---|---|
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

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 ($1 \le M \le 10^6$) — 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.

## Пример

| input.txt | output.txt |
|---|---|
| 4<br>+ 1<br>+ 10<br>-<br>- | 1<br>10 |

```
#include "edx-io.hpp"
using namespace std;


int main() {
```

```
	long N;
	io >> N;
	char action;

	long* stack = new long[N];
	long top = -1;
	long head = 0;

	for (long i = 0; i < N; i++) {
		io >> action;
		if (action == '+') {
			top++;
			io >> stack[top];
		}
		else {
			io << stack[head] << '\n';
			head++;
		}
	}

	return 0;
}
```

| № теста | Результат | Время, с | Память | Размер входного файла | Размер выходного файла |
|---|---|---|---|---|---|
| Max | | 0.156 | 19202048 | 13389454 | 5693807 |
| 1 | OK | 0.000 | 2228224 | 20 | 7 |
| 2 | OK | 0.015 | 2228224 | 11 | 3 |
| 3 | OK | 0.000 | 2228224 | 19 | 6 |
| 4 | OK | 0.015 | 2228224 | 19 | 6 |
| 5 | OK | 0.000 | 2240512 | 96 | 45 |
| 6 | OK | 0.000 | 2240512 | 85 | 56 |
| 7 | OK | 0.015 | 2228224 | 129 | 12 |
| 8 | OK | 0.000 | 2240512 | 131 | 12 |
| 9 | OK | 0.000 | 2240512 | 859 | 538 |
| 10 | OK | 0.015 | 2220032 | 828 | 573 |
| 11 | OK | 0.015 | 2240512 | 1340 | 12 |
| 12 | OK | 0.000 | 2228224 | 1325 | 12 |
| 13 | OK | 0.015 | 2228224 | 8292 | 5589 |
| 14 | OK | 0.015 | 2240512 | 8212 | 5706 |
| 15 | OK | 0.000 | 2228224 | 13298 | 115 |
| 16 | OK | 0.000 | 2228224 | 13354 | 12 |
| 17 | OK | 0.015 | 2244608 | 82372 | 56552 |
| 18 | OK | 0.000 | 2240512 | 82000 | 56993 |
| 19 | OK | 0.015 | 2277376 | 132796 | 1124 |
| 20 | OK | 0.015 | 2265088 | 133914 | 12 |
| 21 | OK | 0.015 | 2838528 | 819651 | 569553 |
| 22 | OK | 0.015 | 2838528 | 819689 | 569681 |
| 23 | OK | 0.000 | 3543040 | 1328670 | 11296 |
| 24 | OK | 0.015 | 3551232 | 1338543 | 12 |
| 25 | OK | 0.156 | 12013568 | 8196274 | 5693025 |
| 26 | OK | 0.140 | 12013568 | 8193816 | 5693807 |
| 27 | OK | 0.062 | 19062784 | 13286863 | 112110 |
| 28 | OK | 0.046 | 19202048 | 13389454 | 10 |
| 29 | OK | 0.046 | 19202048 | 13388564 | 11 |

# 3) Скобочная последовательность

| | |
|---|---|
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

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 $104$ 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.

## Пример

| input.txt | output.txt |
| --- | --- |
| 5<br>()()<br>([])<br>([)]<br>(([]]<br>)( | YES<br>YES<br>NO<br>NO<br>NO |

```
#include "edx-io.hpp"
#include <string>
using namespace std;


string parse(string str, char *stack, long top) {
	for (int j = 0; j < str.size(); j++) {
		switch (str[j]) {
		case '(':
			stack[++top] = str[j];
			continue;
		case '[':
			stack[++top] = str[j];
			continue;
		case ')':
			if ( stack[top] != '(' || top < 0 ) {
				return "NO\n";
			}
			top--;
			continue;
		case ']':
			if (stack[top] != '[' || top < 0  ) {
				return "NO\n";
			}
			top--;
			continue;
		}
	}
	if (top == -1) {
		return "YES\n";
	}
	return "NO\n";
}

int main() {

	long N;
	string str;
	io >> N;
	char* stack = new char[10000];


	for (long i = 0; i < N; i++) {
		long top = -1;
		io >> str;
		io << parse(str, stack, top);
	}

	return 0;
}
```

| № теста | Результат | Время, с | Память | Размер входного файла | Размер выходного файла |
|---|---|---|---|---|---|
| Max | | 0.062 | 6848512 | 5000885 | 2133 |
| 1 | OK | 0.015 | 2236416 | 31 | 22 |
| 2 | OK | 0.015 | 2236416 | 15 | 16 |
| 3 | OK | 0.015 | 2236416 | 68 | 66 |
| 4 | OK | 0.015 | 2240512 | 324 | 256 |
| 5 | OK | 0.000 | 2240512 | 1541 | 1032 |
| 6 | OK | 0.000 | 2240512 | 5880 | 2128 |
| 7 | OK | 0.000 | 2252800 | 50867 | 2129 |
| 8 | OK | 0.000 | 2330624 | 500879 | 2110 |
| 9 | OK | 0.062 | 6848512 | 5000884 | 2120 |
| 10 | OK | 0.046 | 6848512 | 5000885 | 2133 |

## 4) Очередь с минимумом

| | |
|---|---|
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

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 ($1 \le M \le 10^6$) — 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.

## Пример

| input.txt | output.txt |
| --- | --- |
| 7<br>+ 1<br>?<br>+ 10<br>?<br>-<br>?<br>- | 1<br>1<br>10 |

```
#include "edx-io.hpp"
using namespace std;

int main() {
	long N;
	io >> N;
	char action;
	long* queue = new long[N];
	long top = -1;
	long head = 0;
	long *minQueue = new long[N];
	long minTop = -1;
	long minHead = 0;

	for (long i = 0; i < N; i++) {
		io >> action;
		switch (action)
		{
		case '+':
			io >> queue[++top];
			while (minQueue[minTop] > queue[top] && minTop - minHead >= 0 ) {
				minTop--;
			}
			minQueue[++minTop] = queue[top];
			continue;
		case '-':
			if (queue[head] == minQueue[minHead]) {
				minHead++;
			}
			head++;
			continue;
		case '?':
			io << minQueue[minHead] << '\n';
			continue;
		}


	}
	return 0;
}
```

| | | | | | |
|---|---|---|---|---|---|
| 37 | OK | 0.015 | 2609152 | 646300 | 400008 |
| 38 | OK | 0.015 | 2813952 | 588898 | 163890 |
| 39 | OK | 0.000 | 2629632 | 600009 | 175000 |
| 40 | OK | 0.015 | 2818048 | 588898 | 163890 |
| 41 | OK | 0.015 | 2633728 | 600009 | 175000 |
| 42 | OK | 0.109 | 9621504 | 6465010 | 4002151 |
| 43 | OK | 0.062 | 19206144 | 13389342 | 12 |
| 44 | OK | 0.109 | 9617408 | 6462989 | 4000004 |
| 45 | OK | 0.062 | 12210176 | 6388899 | 1888890 |
| 46 | OK | 0.078 | 10317824 | 6500010 | 2000000 |
| 47 | OK | 0.078 | 12210176 | 6388899 | 1888890 |
| 48 | OK | 0.062 | 10321920 | 6500010 | 2000000 |
| 49 | OK | 0.093 | 19202048 | 13388086 | 12 |
| 50 | OK | 0.015 | 2220032 | 55 | 16 |
| 51 | OK | 0.015 | 2224128 | 705 | 225 |
| 52 | OK | 0.031 | 2248704 | 6506 | 2000 |
| 53 | OK | 0.000 | 2285568 | 65007 | 20000 |
| 54 | OK | 0.015 | 2875392 | 650008 | 200000 |
| 55 | OK | 0.078 | 12496896 | 6675213 | 2000000 |
| 56 | OK | 0.000 | 2224128 | 117 | 12 |
| 57 | OK | 0.015 | 2224128 | 1327 | 12 |
| 58 | OK | 0.000 | 2248704 | 13417 | 12 |
| 59 | OK | 0.000 | 2306048 | 133845 | 12 |
| 60 | OK | 0.000 | 3952640 | 1339319 | 12 |
| 61 | OK | 0.062 | 23199744 | 13388955 | 12 |

## 5)Quack

| | |
|---|---|
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

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 $\alpha$ и $\beta$ подразумеваются некие абстрактные временные переменные):

| | |
|---|---|
| + | Сложение: get $\alpha$, get $\beta$, put $(\alpha+\beta) \bmod 65536$ |
| - | Вычитание: get $\alpha$, get $\beta$, put $(\alpha-\beta) \bmod 65536$ |
| * | Умножение: get $\alpha$, get $\beta$, put $(\alpha\cdot\beta) \bmod 65536$ |
| / | Целочисленное деление: get $\alpha$, get $\beta$, put $\alpha \operatorname{div} \beta$ (будем считать, что $\alpha \operatorname{div} 0 = 0$) |
| % | Взятие по модулю: get $\alpha$, get $\beta$, put $\alpha \bmod \beta$ (будем считать, что $\alpha \bmod 0 = 0$) |
| >[register] | Положить в регистр: get $\alpha$, установить значение [register] в $\alpha$ |
| <[register] | Взять из регистра: put значение [register] |
| P | Напечатать: get $\alpha$, вывести $\alpha$ 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 $\alpha$, вывести символ с ASCII-кодом $\alpha \bmod 256$ в стандартный поток вывода |
| C[register] | Вывести регистр как символ: вывести символ с ASCII-кодом $\alpha \bmod 256$ (где $\alpha$ — 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 105 шагов. Программа содержит не менее одной и не более 105 инструкций. **Метки имеют длину от 1 до 10 и состоят из цифр и латинских букв.**

```
#include <fstream>
#include <queue>
#include <string>
#include <map>
using namespace std;


int main() {
	ifstream input("input.txt");
	ofstream output("output.txt");

	queue<unsigned short> MainQueue = {};
	map<char, unsigned short> registers = {
		{'a', 0},
		{'b', 0},
		{'c', 0},
		{'d', 0},
		{'e', 0},
		{'f', 0},
		{'g', 0},
		{'h', 0},
		{'i', 0},
		{'j', 0},
		{'k', 0},
		{'l', 0},
		{'m', 0},
		{'n', 0},
		{'o', 0},
		{'p', 0},
		{'q', 0},
		{'r', 0},
		{'s', 0},
		{'t', 0},
		{'u', 0},
		{'v', 0},
		{'w', 0},
		{'x', 0},
		{'y', 0},
		{'z', 0}
	};
	map <string, long> labels;
	string *commands = new string[100001];
	//Получаем список команд
	long N = -1;
	while (!input.eof()) {
		input >> commands[++N];
		//Запоминаем все метки
		if (commands[N][0] == ':') {
			labels.insert(pair<string, long>(commands[N].substr(1), N));
		}
	}
	if (commands[N].empty()) {
		N--;
	}
	input.close();
	//Указатель на команду в списке
	long P = 0;
	unsigned short a, b;

	do {
		switch (commands[P][0])
		{
		case '+':
			a = MainQueue.front();
```

```
		MainQueue.pop();
		b = MainQueue.front();
		MainQueue.pop();
		MainQueue.push((a + b) % 65536);
		P++;
		break;
	case '-':
		a = MainQueue.front();
		MainQueue.pop();
		b = MainQueue.front();
		MainQueue.pop();
		MainQueue.push((a - b) % 65536);
		P++;
		break;
	case '*':
		a = MainQueue.front();
		MainQueue.pop();
		b = MainQueue.front();
		MainQueue.pop();
		MainQueue.push((a * b) % 65536);
		P++;
		break;
	case '/':
		a = MainQueue.front();
		MainQueue.pop();
		b = MainQueue.front();
		MainQueue.pop();
		if (b == 0) {
			MainQueue.push(0);
		}
		else {
			MainQueue.push(a / b);
		}
		P++;
		break;
	case '%':
		a = MainQueue.front();
		MainQueue.pop();
		b = MainQueue.front();
		MainQueue.pop();
		if (b == 0) {
			MainQueue.push(0);
		}
		else {
			MainQueue.push(a % b);
		}
		P++;
		break;
	case '>':
		a = MainQueue.front();
		MainQueue.pop();
		registers[commands[P][1]] = a;
		P++;
		break;
	case '<':
		MainQueue.push(registers[commands[P][1]]);
		P++;
		break;
	case 'P':
		if (commands[P].size() == 1) {
			output << MainQueue.front() << '\n';
			MainQueue.pop();
		}
		else {
			output << registers[commands[P][1]] << '\n';
		}
		P++;
		break;
	case 'C':
		if (commands[P].size() == 1) {
			output << (char)(MainQueue.front() % 256);
```

```
                        MainQueue.pop();
                }
                else {
                        output << (char)(registers[commands[P][1]] % 256);
                }
                P++;
                break;
        case ':':
                P++;
                break;
        case 'J':
                P = labels[commands[P].substr(1)] + 1;
                break;
        case 'Z':
                if (registers[commands[P][1]] == 0) {
                        P = labels[commands[P].substr(2)] + 1;
                }
                else {
                        P++;
                }
                break;
        case 'E':
                if (registers[commands[P][1]] == registers[commands[P][2]]) {
                        P = labels[commands[P].substr(3)] + 1;
                }
                else {
                        P++;
                }
                break;
        case 'G':
                if (registers[commands[P][1]] > registers[commands[P][2]]) {
                        P = labels[commands[P].substr(3)] + 1;
                }
                else {
                        P++;
                }
                break;
        case 'Q':
                P = N;
                P++;
                break;
        default:
                MainQueue.push(stoi(commands[P]));
                P++;
        }

    } while (P <= N);
    output.close();

    return 0;
}
```

| № теста | Результат | Время, с | Память | Размер входного файла | Размер выходного файла |
|---|---|---|---|---|---|
| Max | | 0.078 | 9293824 | 1349803 | 250850 |
| 1 | OK | 0.000 | 5591040 | 69 | 6 |
| 2 | OK | 0.000 | 5574656 | 232 | 232 |
| 3 | OK | 0.015 | 5566464 | 3 | 0 |
| 4 | OK | 0.015 | 5591040 | 100 | 19 |
| 5 | OK | 0.031 | 5574656 | 56 | 58890 |
| 6 | OK | 0.015 | 5591040 | 67 | 30000 |
| 7 | OK | 0.015 | 5578752 | 67 | 30000 |
| 8 | OK | 0.015 | 5570560 | 55 | 30000 |
| 9 | OK | 0.000 | 5586944 | 461 | 62 |
| 10 | OK | 0.015 | 5566464 | 11235 | 21 |
| 11 | OK | 0.015 | 5574656 | 23748 | 42 |
| 12 | OK | 0.015 | 5574656 | 66906 | 9136 |
| 13 | OK | 0.015 | 5586944 | 7332 | 993 |
| 14 | OK | 0.000 | 5586944 | 4611 | 632 |
| 15 | OK | 0.015 | 5574656 | 37968 | 7332 |
| 16 | OK | 0.000 | 5586944 | 14 | 3 |
| 17 | OK | 0.015 | 5574656 | 70 | 14 |
| 18 | OK | 0.015 | 5586944 | 350 | 70 |
| 19 | OK | 0.015 | 5586944 | 1750 | 350 |
| 20 | OK | 0.000 | 5582848 | 8750 | 1750 |
| 21 | OK | 0.000 | 5586944 | 43750 | 8750 |
| 22 | OK | 0.031 | 5582848 | 218750 | 43750 |
| 23 | OK | 0.000 | 5582848 | 34606 | 4867 |
| 24 | OK | 0.046 | 5787648 | 683180 | 7 |
| 25 | OK | 0.046 | 5767168 | 683102 | 0 |
| 26 | OK | 0.078 | 9293824 | 1349803 | 0 |
| 27 | OK | 0.078 | 5591040 | 491572 | 247791 |
| 28 | OK | 0.062 | 5591040 | 491488 | 249618 |
| 29 | OK | 0.062 | 5595136 | 491600 | 249600 |
| 30 | OK | 0.078 | 5595136 | 491502 | 250850 |
| 31 | OK | 0.078 | 5591040 | 491416 | 249477 |
| 32 | OK | 0.078 | 5586944 | 491520 | 250262 |
| 33 | OK | 0.062 | 5591040 | 491317 | 246859 |
| 34 | OK | 0.078 | 5595136 | 491514 | 248199 |
| 35 | OK | 0.078 | 5586944 | 491557 | 249601 |