Министерство образования и науки Российской Федерации
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Дисциплина «**Алгоритмы и структуры данных**»

ОТЧЁТ

по лабораторным работам за 4 неделю

Студент **Пастухов К.А. группы P3218**

Санкт-Петербург
2018 г.

Стек

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

Выполнение

```
def main():
    with open('input.txt', "r") as file:
        commands = int(file.readline())
        stack = []
        res = []
        for i in range(1, commands + 1):
            data = [x for x in file.readline().strip().split(" ")]
            if data[0] == "+":
                stack.append(int(data[1]))
            else:
                res.append(stack.pop())
    with open("output.txt", "w") as res_file:
        res_file.write('\n'.join(str(d) for d in res))


if __name__ == '__main__':
    main()
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.109	73306112	13389454	5693805
1	OK	0.046	9293824	33	8
2	OK	0.046	9252864	11	1
3	OK	0.062	9281536	19	4
4	OK	0.031	9338880	19	4
5	OK	0.062	9297920	19	4
6	OK	0.031	9326592	96	43
7	OK	0.031	9211904	85	54
8	OK	0.031	9240576	129	9
9	OK	0.062	9269248	131	10
10	OK	0.031	9338880	859	538
11	OK	0.031	9351168	828	571
12	OK	0.046	9277440	1340	9
13	OK	0.078	9285632	1325	10
14	OK	0.046	9359360	8292	5588
15	OK	0.046	9277440	8212	5704
16	OK	0.031	9351168	13298	109
17	OK	0.031	9326592	13354	10
18	OK	0.046	9957376	82372	56546
19	OK	0.046	9932800	82000	56991
20	OK	0.046	9830400	132796	1132
21	OK	0.046	9711616	133914	9
22	OK	0.171	15663104	819651	569555
23	OK	0.140	15925248	819689	569679
24	OK	0.140	14336000	1328670	11292
25	OK	0.140	14831616	1338543	9
26	OK	1.109	73306112	8196274	5693033
27	OK	1.109	72867840	8193816	5693805
28	OK	1.062	56553472	13286863	112018
29	OK	1.062	56176640	13389454	9
30	OK	1.078	56201216	13388564	9

Очередь

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

Выполнение

```
from collections import deque


def main():
    with open('input.txt', "r") as file:
        commands = int(file.readline())
        queue = deque([])
        res = []
        for i in range(1, commands + 1):
            data = [x for x in file.readline().strip().split(" ")]
            if data[0] == "+":
                queue.append(int(data[1]))
            else:
                res.append(queue.popleft())
    with open("output.txt", "w") as res_file:
        res_file.write('\n'.join(str(d) for d in res))


if __name__ == '__main__':
    main()
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		2.000	75677696	13389454	5693805
1	OK	0.046	9478144	20	5
2	OK	0.031	9515008	11	1
3	OK	0.078	9482240	19	4
4	OK	0.046	9486336	19	4
5	OK	0.046	9523200	96	43
6	OK	0.062	9523200	85	54
7	OK	0.031	9469952	129	10
8	OK	0.031	9498624	131	10
9	OK	0.046	9465856	859	536
10	OK	0.046	9510912	828	571
11	OK	0.031	9523200	1340	10
12	OK	0.031	9515008	1325	10
13	OK	0.031	9560064	8292	5587
14	OK	0.046	9547776	8212	5704
15	OK	0.046	9605120	13298	113
16	OK	0.046	9584640	13354	10
17	OK	0.062	10088448	82372	56550
18	OK	0.062	10145792	82000	56991
19	OK	0.046	9875456	132796	1122
20	OK	0.078	9916416	133914	10
21	OK	0.218	15937536	819651	569551
22	OK	0.250	17276928	819689	569679
23	OK	0.218	13856768	1328670	11294
24	OK	0.218	13737984	1338543	10
25	OK	2.000	72151040	8196274	5693023
26	OK	1.968	75677696	8193816	5693805
27	OK	1.859	51437568	13286863	112108
28	OK	1.890	50704384	13389454	8
29	OK	1.906	50741248	13388564	9

Скобочная последовательность

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

Выполнение

```
def main():
    with open('input.txt', "r") as file:
        commands = int(file.readline())
        res = []
        for i in range(1, commands + 1):
            data = file.readline().strip()
            res.append(check_brackets(data))
    with open("output.txt", "w") as res_file:
        for i in res:
            res_file.write("YES\n" if i else "NO\n")


def check_brackets(text):
    opening = "(["
    closing = ")]"
    stack = []
    for character in text:
        if character in opening:
            stack.append(opening.index(character))
            print(stack)
        elif character in closing:
            if stack and stack[-1] == closing.index(character):
                stack.pop()
            else:
                return False

    return (not stack)


if __name__ == '__main__':
    main()
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.593	10776576	5000885	2133
1	OK	0.031	9011200	31	22
2	OK	0.046	8933376	15	16
3	OK	0.031	8937472	68	66
4	OK	0.031	8941568	324	256
5	OK	0.031	8990720	1541	1032
6	OK	0.031	9007104	5880	2128
7	OK	0.046	9031680	50867	2129
8	OK	0.187	9232384	500879	2110
9	OK	1.531	10362880	5000884	2120
10	OK	1.593	10776576	5000885	2133

Очередь с минимумом

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.

Выполнение

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace Week4_4 {
    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();
        }
    }
}
```

Quack

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

Выполнение

```
import string
import queue
from edx_io import edx_io

registers = {}
queue = queue.Queue()
labels = {}
p = 0
q = False


def put(v):
    queue.put(v % 65536, False)


def get():
    return queue.get(False)
```

```
def jump(lbl):
    global p
    p = labels[lbl]


def execute(cmd, io):
    c = cmd[0]
    if c == '+':
        put(get() + get())
    elif c == '-':
        a = get()
        put(a - get())
    elif c == '*':
        put(get() * get())
    elif c == '/':
        a = get()
        b = get()
        put(0 if b == 0 else a // b)
    elif c == '%':
        a = get()
        b = get()
        put(0 if b == 0 else a % b)
    elif c == '>':
        registers[cmd[1]] = get()
    elif c == '<':
        put(registers[cmd[1]])
    elif c == 'P':
        if len(cmd) == 1:
            io.writeln(get())
        else:
            io.writeln(registers[cmd[1]])
    elif c == 'C':
        if len(cmd) == 1:
            io.write(chr(get() % 256))
        else:
            io.write(chr(registers[cmd[1]] % 256))
    elif c == ':':
        labels[cmd[1:]] = p
    elif c == 'J':
        jump(cmd[1:])
    elif c == 'Z':
        if registers[cmd[1]] == 0:
            jump(cmd[2:])
    elif c == 'E':
        if registers[cmd[1]] == registers[cmd[2]]:
            jump(cmd[3:])
    elif c == 'G':
        if registers[cmd[1]] > registers[cmd[2]]:
            jump(cmd[3:])
    elif c == 'Q':
        global q
        q = True
    else:
        put(int(cmd))


with edx_io() as io:
    tokens = [t.decode() for t in io.all_tokens()]
```

```
print(tokens)
labels = {label[1:]: i for i, label in enumerate(tokens) if label[0] == ':'}
print(labels)
registers = dict.fromkeys(string.ascii_lowercase, 0)

print(registers)
lt = len(tokens)

while p < lt and not q:
    execute(tokens[p], io)
    p += 1
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.843	30691328	1349803	213100
1	OK	0.078	11509760	69	5
2	OK	0.062	11583488	232	174
3	OK	0.062	11640832	3	0
4	OK	0.062	11513856	100	14
5	OK	0.375	11616256	56	48890
6	OK	0.218	11538432	67	25000
7	OK	0.218	11571200	67	25000
8	OK	0.234	11526144	55	25000
9	OK	0.062	11558912	461	60
10	OK	0.109	11591680	11235	21
11	OK	0.109	11653120	23748	42
12	OK	0.171	12455936	66906	8905
13	OK	0.078	11698176	7332	954
14	OK	0.078	11608064	4611	602
15	OK	0.140	12079104	37968	5424
16	OK	0.062	11522048	14	2
17	OK	0.062	11538432	70	10
18	OK	0.078	11456512	350	50
19	OK	0.062	11517952	1750	250