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 по лабораторной работе Неделя №4

OpenEdu

Группа: P3217
Выполнил: Минин Александр

г. Санкт-Петербург
2018г.

Задача №1

Условие

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

```
package week4;

import mooc.EdxIO;

public class Week4_1 {
    private static EdxIO edxIO;

    private static int[] stackArray = new int[(int) Math.pow(10, 6)];
    private static int stackPointer = 0;

    public static void main(String[] args) {
        edxIO = EdxIO.create();

        int n = edxIO.nextInt();
        for (int i = 0; i < n; i++) {
            switch (edxIO.nextChar()) {
                case '+':
                    stackArray[stackPointer] = edxIO.nextInt();
                    stackPointer++;
                    break;
                case '-':
                    edxIO.println(stackArray[stackPointer - 1]);
                    stackPointer--;
                    break;
            }
        }

        edxIO.close();
    }
}
```

Результаты

№ теста	Результат	Время, мс	Память	Размер входного файла	Размер выходного файла
Max		343	41709568	13389454	5693807
1	OK	156	25583616	33	10
2	OK	109	25608192	11	3
3	OK	140	25563136	19	6
4	OK	156	25530368	19	6
5	OK	109	25624576	19	6
6	OK	109	25567232	96	45
7	OK	109	25600000	85	56
8	OK	156	25579520	129	11
9	OK	140	25571328	131	12
10	OK	140	25559040	859	540
11	OK	109	25616384	828	573
12	OK	125	25636864	1340	11
13	OK	93	25563136	1325	12
14	OK	140	25731072	8292	5590
15	OK	140	25702400	8212	5706
16	OK	125	25767936	13298	111
17	OK	156	25784320	13354	12
18	OK	125	26652672	82372	56548
19	OK	125	27312128	82000	56993
20	OK	109	27070464	132796	1134
21	OK	109	26849280	133914	11
22	OK	234	29581312	819651	569557
23	OK	203	28934144	819689	569681
24	OK	187	28467200	1328670	11294
25	OK	187	28475392	1338543	11
26	OK	343	36327424	8196274	5693035
27	OK	312	36438016	8193816	5693807
28	OK	296	41709568	13286863	112020
29	OK	218	40398848	13389454	11
30	OK	218	40464384	13388564	11

Задача №2

Условие

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

```
package week4;

import mooc.EdxIO;

public class Week4_2 {
    private static EdxIO edxIO;

    private static int[] queueArray = new int[(int) Math.pow(10, 6)];
    private static int queueTailPointer = 0;
    private static int queueHeadPointer = 0;

    public static void main(String[] args) {
        edxIO = EdxIO.create();

        int n = edxIO.nextInt();
        for (int i = 0; i < n; i++) {
            switch (edxIO.nextChar()) {
                case '+':
                    queueArray[queueTailPointer] = edxIO.nextInt();
                    queueTailPointer = (queueTailPointer + 1) % queueArray.length;
                    break;
                case '-':
                    edxIO.println(queueArray[queueHeadPointer]);
                    queueHeadPointer = (queueHeadPointer + 1) % queueArray.length;
                    break;
            }
        }

        edxIO.close();
    }
}
```

Результаты

№ теста	Результат	Время, мс	Память	Размер входного файла	Размер выходного файла
Max		359	41656320	13389454	5693807
1	OK	125	25550848	20	7
2	OK	109	25583616	11	3
3	OK	156	25616384	19	6
4	OK	109	25571328	19	6
5	OK	109	25546752	96	45
6	OK	140	25595904	85	56
7	OK	109	25563136	129	12
8	OK	125	25550848	131	12
9	OK	171	25595904	859	538
10	OK	156	25583616	828	573
11	OK	109	25616384	1340	12
12	OK	187	25583616	1325	12
13	OK	156	25739264	8292	5589
14	OK	156	25681920	8212	5706
15	OK	140	25825280	13298	115
16	OK	140	25763840	13354	12
17	OK	109	26669056	82372	56552
18	OK	109	27193344	82000	56993
19	OK	171	27074560	132796	1124
20	OK	109	26906624	133914	12
21	OK	218	29564928	819651	569553
22	OK	187	29016064	819689	569681
23	OK	156	28512256	1328670	11296
24	OK	171	28422144	1338543	12
25	OK	328	36491264	8196274	5693025
26	OK	359	36548608	8193816	5693807
27	OK	281	41656320	13286863	112110
28	OK	265	40464384	13389454	10
29	OK	265	40534016	13388564	11

Задача №3

Условие

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

```
package week4;

import mooc.EdxIO;

public class Week4_3 {
   private static EdxIO edxIO;

   private static char[] stackArray = new char[5000];

   private static int stackPointer = 0;

   public static char getStartingBracket(char closingBracket) {
      switch (closingBracket) {
         case ']':
            return '[';
         case ')':
            return '(';
         default:
            throw new IllegalStateException();
      }
   }
```

```
    public static void main(String[] args) {
        edxIO = EdxIO.create();

        int n = edxIO.nextInt();
        char ch;
        byte[] sequence;
        boolean isValid = true;
        for (int i = 0; i < n; i++) {
            sequence = edxIO.nextBytes();
            for (int k = 0; k < sequence.length && isValid; k++) {
                ch = (char) sequence[k];
                switch (ch) {
                    case '(':
                    case '[':
                        if (stackPointer > stackArray.length - 1) {
                            // it's impossible that later we meet more than stack.Array.length closing
brackets
                            isValid = false;
                            break;
                        }
                        stackArray[stackPointer] = ch;
                        stackPointer++;
                        break;
                    case ')':
                    case ']':
                        if (stackPointer < 1 || getStartingBracket(ch) != stackArray[stackPointer - 1]) {
                            isValid = false;
                            break;
                        }
                        stackPointer--;
                        break;
                    default:
                        throw new IllegalStateException();
                }
            }
            edxIO.println(isValid && stackPointer == 0 ? "YES" : "NO");
            stackPointer = 0;
            isValid = true;
        }

        edxIO.close();
    }
}
```

Результаты

№ теста	Результат	Время, мс	Память	Размер входного файла	Размер выходного файла
Max		218	33361920	5000885	2133
1	OK	125	21553152	31	22
2	OK	125	21569536	15	16
3	OK	156	21598208	68	66
4	OK	140	21614592	324	256
5	OK	125	21610496	1541	1032

6	OK	140	21729280	5880	2128
7	OK	125	21843968	50867	2129
8	OK	140	24981504	500879	2110
9	OK	218	33361920	5000884	2120
10	OK	203	33255424	5000885	2133

Задача №4

Условие

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

```
package week4;

import mooc.EdxIO;

public class Week4_4 {

    public static void main(String[] args) {
        EdxIO edxIO = EdxIO.create();

        int n = edxIO.nextInt();
        QueueWithMinimal<Integer> queueWithMinimal = new QueueWithMinimal<>((int)
Math.pow(10, 6));

        for (int i = 0; i < n; i++) {
            switch (edxIO.nextChar()) {
                case '+':
                    queueWithMinimal.push(edxIO.nextInt());
                    break;
                case '-':
                    queueWithMinimal.pop();
                    break;
                case '?':
                    edxIO.println(queueWithMinimal.getMin());
```

```
                break;
            }
        }

        edxIO.close();
    }

    private static class StackWithMinimal<T extends Comparable> {

        private Object[] array;

        private Object[] minArray;

        private int stackPointer = 0;

        public StackWithMinimal(int initialCapacity) {
            array = new Object[initialCapacity];
            minArray = new Object[initialCapacity];
        }

        public int getSize() {
            return stackPointer;
        }

        public void push(T el) {
            if (el == null) {
                throw new IllegalArgumentException("El can not be null");
            }

            array[stackPointer] = el;

            Object min = el;
            if (getSize() > 0 && el.compareTo(minArray[stackPointer - 1]) >= 0) {
                min = minArray[stackPointer - 1];
            }
            minArray[stackPointer] = min;

            stackPointer++;
        }

        public T pop() {
            T result = peek();
            stackPointer--;
            array[stackPointer] = null;
            minArray[stackPointer] = null;
            return result;
        }

        public T peek() {
            if (getSize() == 0) {
                throw new IllegalStateException("No elements in stack");
            }
            return (T) array[stackPointer - 1];
        }

        public T getMin() {
            if (getSize() == 0) {
                throw new IllegalStateException("No elements in stack");
            }
            return (T) minArray[stackPointer - 1];
```

```
        }

        public boolean isEmpty() {
            return getSize() == 0;
        }
    }

    public static class QueueWithMinimal<T extends Comparable> {

        private StackWithMinimal<T> s1;
        private StackWithMinimal<T> s2;

        public QueueWithMinimal(int initialCapacity) {
            s1 = new StackWithMinimal<>(initialCapacity);
            s2 = new StackWithMinimal<>(initialCapacity);
        }

        public int getSize() {
            return s1.getSize() + s2.getSize();
        }

        public void push(T el) {
            s1.push(el);
        }

        public T pop() {
            if (getSize() == 0) {
                throw new IllegalStateException("Queue is empty");
            }

            if (!s2.isEmpty()) {
                return s2.pop();
            }

            while (s1.getSize() != 0) {
                s2.push(s1.pop());
            }

            return s2.pop();
        }

        public T getMin() {
            if (getSize() == 0) {
                throw new IllegalStateException("Queue is empty");
            }

            if (s1.isEmpty()) {
                return s2.getMin();
            } else if (s2.isEmpty()) {
                return s1.getMin();
            } else {
                if ((s1.getMin().compareTo(s2.getMin()) > 0)) {
                    return s2.getMin();
                } else {
                    return s1.getMin();
                }
            }
        }
    }
}
```

Результаты

№ теста	Результат	Время, мс	Память	Размер входного файла	Размер выходного файла
Max		593	99524608	13389342	4002151
1	OK	125	37617664	29	10
2	OK	125	37580800	11	3
3	OK	171	37625856	22	6
4	OK	125	37621760	22	6
5	OK	140	37634048	36	9
6	OK	125	37580800	48	12
7	OK	125	37617664	76	35
8	OK	125	37584896	129	12
9	OK	109	37617664	67	48
10	OK	125	37621760	44	9
11	OK	140	37572608	45	9
12	OK	171	37560320	44	9
13	OK	125	37597184	45	9
14	OK	125	37609472	721	384
15	OK	125	37605376	1340	12
16	OK	140	37588992	640	407
17	OK	171	37609472	445	90
18	OK	125	37634048	456	100
19	OK	125	37576704	445	90
20	OK	125	37609472	456	100
21	OK	171	37707776	6616	3812
22	OK	140	37777408	13389	12
23	OK	140	37699584	6461	4008
24	OK	140	37761024	4896	1140
25	OK	171	37683200	5007	1250
26	OK	140	37769216	4896	1140
27	OK	140	37740544	5007	1250
28	OK	156	38756352	64907	39589
29	OK	140	39137280	133814	12
30	OK	203	39485440	64675	39996
31	OK	171	39608320	53897	13890

32	OK	171	39587840	55008	15000
33	OK	171	39641088	53897	13890
34	OK	171	39460864	55008	15000
35	OK	218	41115648	645271	404305
36	OK	187	42229760	1338956	12
37	OK	250	41373696	646300	400008
38	OK	203	40841216	588898	163890
39	OK	187	40894464	600009	175000
40	OK	234	40927232	588898	163890
41	OK	171	40886272	600009	175000
42	OK	343	52817920	6465010	4002151
43	OK	468	99266560	13389342	12
44	OK	375	52260864	6462989	4000004
45	OK	390	54808576	6388899	1888890
46	OK	343	54927360	6500010	2000000
47	OK	375	54763520	6388899	1888890
48	OK	343	54947840	6500010	2000000
49	OK	484	99336192	13388086	12
50	OK	140	37634048	55	16
51	OK	140	37613568	705	225
52	OK	156	37752832	6506	2000
53	OK	187	39145472	65007	20000
54	OK	234	41095168	650008	200000
55	OK	328	55582720	6675213	2000000
56	OK	140	37601280	117	12
57	OK	140	37625856	1327	12
58	OK	156	37806080	13417	12
59	OK	140	39469056	133845	12
60	OK	250	42491904	1339319	12
61	OK	593	99524608	13388955	12

Задача №5

Условие

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

Решение

```
package week4;

import mooc.EdxIO;

import java.util.ArrayList;
import java.util.HashMap;

public class Week4_5 {
    private static EdxIO edxIO;

    /**
     * BEGIN Queue
     */
    private static int[] queueArray = new int[(int) Math.pow(10, 5)];
    private static int queueTailPointer = 0;
    private static int queueHeadPointer = 0;

    private static int pop() {
        int result = queueArray[queueHeadPointer];
        queueHeadPointer = (queueHeadPointer + 1) % queueArray.length;
```

```
    return result;
}

private static void push(int el) {
    queueArray[queueTailPointer] = Integer.remainderUnsigned(el, 65536);
    queueTailPointer = (queueTailPointer + 1) % queueArray.length;
}

/**
 * END Queue
 */

/**
 * BEGIN VM Structure
 */
private static int[] registers = new int['z' - 'a' + 1];
private static ArrayList<byte[]> code = new ArrayList();
private static HashMap<String, Integer> labels = new HashMap<String, Integer>();

/**
 * END VM Structure
 */

public static void main(String[] args) {
    edxIO = EdxIO.create();

    byte[] ins = edxIO.nextBytes();
    int i = 0;
    do {
        code.add(ins);
        if (ins[0] == ':') {
            labels.put(new String(ins).substring(1), i);
        }
        i++;
        ins = edxIO.nextBytes();

    } while (ins != null);

    int ip = 0;
    while (ip < code.size()) {
        ins = code.get(ip);
        switch (ins[0]) {
            case '+':
                push(pop() + pop());
                break;
            case '-':
                push(pop() - pop());
                break;
            case '*':
                push(pop() * pop());
                break;
            case '/':
                int a = pop();
                int b = pop();
                push(b == 0 ? 0 : a / b);
                break;
            case '%':
                a = pop();
                b = pop();
                push(b == 0 ? 0 : a % b);
```

```
                break;
            case '>':
                registers[ins[1] - 'a'] = pop();
                break;
            case '<':
                push(registers[ins[1] - 'a']);
                break;
            case 'P':
                edxIO.println(ins.length == 1 ? pop() : registers[ins[1] - 'a']);
                break;
            case 'C':
                edxIO.print((char) (ins.length == 1 ? pop() % 256 : registers[ins[1] - 'a'] % 256));
                break;
            case ':':
                break;
            case 'J':
                ip = labels.get(new String(ins).substring(1));
                break;
            case 'Z':
                if (registers[ins[1] - 'a'] == 0) {
                    ip = labels.get(new String(ins).substring(2));
                }
                break;
            case 'E':
                if (registers[ins[1] - 'a'] == registers[ins[2] - 'a']) {
                    ip = labels.get(new String(ins).substring(3));
                }
                break;
            case 'G':
                if (registers[ins[1] - 'a'] > registers[ins[2] - 'a']) {
                    ip = labels.get(new String(ins).substring(3));
                }
                break;
            case 'Q':
                edxIO.close();
                return;
            default:
                int val = 0;
                for (int k = 0; k < ins.length; k++) {
                    val += (ins[k] - '0') * Math.pow(10, ins.length - k - 1);
                }
                push(val);
        }
        ip++;
    }


    edxIO.close();
    return;
  }

}
```

Результаты

№ теста	Результат	Время, мс	Память	Размер входного файла	Размер выходного файла
Max		312	44621824	1349803	250850

1	OK	140	22048768	69	6
2	OK	156	22044672	232	232
3	OK	125	21983232	3	0
4	OK	140	21995520	100	19
5	OK	203	27013120	56	58890
6	OK	156	23535616	67	30000
7	OK	140	23539712	67	30000
8	OK	187	23699456	55	30000
9	OK	109	21979136	461	62
10	OK	156	22376448	11235	21
11	OK	125	23265280	23748	42
12	OK	156	24440832	66906	9136
13	OK	125	22216704	7332	993
14	OK	109	22142976	4611	632
15	OK	140	24104960	37968	7332
16	OK	125	22016000	14	3
17	OK	125	21991424	70	14
18	OK	109	21966848	350	70
19	OK	109	22052864	1750	350
20	OK	140	22233088	8750	1750
21	OK	156	24068096	43750	8750
22	OK	187	27586560	218750	43750
23	OK	125	23891968	34606	4867
24	OK	250	29544448	683180	7
25	OK	234	29052928	683102	0
26	OK	312	44621824	1349803	0
27	OK	234	30941184	491572	247791
28	OK	234	30871552	491488	249618
29	OK	250	30846976	491600	249600
30	OK	234	30937088	491502	250850
31	OK	234	30842880	491416	249477
32	OK	281	31145984	491520	250262
33	OK	250	30846976	491317	246859
34	OK	250	30875648	491514	248199
35	OK	234	30482432	491557	249601