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 №8

OpenEdu

Группа: P3217
Выполнил: Минин Александр

г. Санкт-Петербург
2019г.

Задача №1

Условие

Реализуйте множество с операциями «добавление ключа», «удаление ключа», «проверка существования ключа».

Формат входного файла

В первой строке входного файла находится строго положительное целое число операций N, не превышающее $5 \cdot 10^5$. В каждой из последующих N строк находится одна из следующих операций:

- `A` x — добавить элемент x в множество. Если элемент уже есть в множестве, то ничего делать не надо.
- `D` x — удалить элемент x. Если элемента x нет, то ничего делать не надо.
- `?` x — если ключ x есть в множестве, выведите `Y`, если нет, то выведите `N`.

Аргументы указанных выше операций — целые числа, не превышающие по модулю 10^{18}.

Формат выходного файла

Выведите последовательно результат выполнения всех операций «`?`». Следуйте формату выходного файла из примера.

Решение

```
package week8;

import mooc.EdxIO;

import java.util.HashSet;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Set;

public class Week8_1 {
    private static EdxIO edxIO;

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

        int n = edxIO.nextInt();
        Set<Long> set = new HashSet<Long>(Integer.MAX_VALUE/100, 0.75f);
        for (int i = 0; i < n; i++) {
            switch (edxIO.nextChar()) {
                case 'A':
                    Long val = edxIO.nextLong();
                    set.add(val);
                    break;
                case 'D':
                    set.remove(edxIO.nextLong());
                    break;
                case '?':
                    edxIO.println(set.contains(edxIO.nextLong()) ? 'Y' : 'N');
                    break;
            }
```

```
        }

        edxIO.close();

    }
}
```

Результаты

№ теста	Результат	Время, мс	Память	Размер входного файла	Размер выходного файла
Max		843	222523392	11189636	501237
1	OK	203	156340224	43	9
2	OK	156	21577728	8	3
3	OK	234	156348416	51	12
4	OK	187	156364800	542	99
5	OK	187	156352512	618	54
6	OK	218	156495872	5451	1038
7	OK	187	156463104	6436	957
8	OK	218	156532736	13382	957
9	OK	187	156622848	22394	981
10	OK	203	156618752	7030	465
11	OK	203	156647424	7020	411
12	OK	187	157515776	63829	10002
13	OK	281	158916608	80339	4947
14	OK	250	158687232	80203	5034
15	OK	234	159846400	545113	100323
16	OK	312	159862784	639485	99282
17	OK	281	162443264	738870	99558
18	OK	328	163864576	1338668	99636
19	OK	281	164507648	2237627	99540
20	OK	281	164507648	903052	50202
21	OK	265	164134912	902843	49536
22	OK	359	166711296	2725205	501237
23	OK	375	167346176	3196877	499713
24	OK	437	178925568	3694712	501051
25	OK	500	185368576	6694340	500355
26	OK	515	189751296	11189636	500040

27	OK	421	189091840	4902931	249012
28	OK	453	188600320	4902757	250305
29	OK	843	222523392	9687139	300000
30	OK	406	193736704	9687570	300000
31	OK	390	192524288	8000008	300000
32	OK	453	197259264	11000008	150000

Задача №2

Условие

Формат входного файла

В первой строке входного файла находится строго положительное целое число операций N, не превышающее $5 \cdot 10^5$. В каждой из последующих N строк находится одна из следующих операций:

- `get` x — если ключ x есть в множестве, выведите соответствующее ему значение, если нет, то выведите `<none>`.
- `prev` x — вывести значение, соответствующее ключу, находящемуся в ассоциативном массиве, который был вставлен позже всех, но до x, или `<none>`, если такого нет или в массиве нет x.
- `next` x — вывести значение, соответствующее ключу, находящемуся в ассоциативном массиве, который был вставлен раньше всех, но после x, или `<none>`, если такого нет или в массиве нет x .
- `put` x y — поставить в соответствие ключу x значение y. При этом следует учесть, что:
  - если, независимо от предыстории, этого ключа на момент вставки в массиве не было, то он считается только что вставленным и оказывается самым последним среди добавленных элементов — то есть, вызов `next` с этим же ключом сразу после выполнения текущей операции `put` должен вернуть `<none>`;
  - если этот ключ уже есть в массиве, то значение необходимо изменить, и в этом случае ключ не считается вставленным еще раз, то есть, не меняет своего положения в порядке добавленных элементов.
- `delete` x — удалить ключ x. Если ключа x в ассоциативном массиве нет, то ничего делать не надо.

Ключи и значения — строки из латинских букв длиной не менее одного и не более 20 символов.

Формат выходного файла

Выведите последовательно результат выполнения всех операций `get`, `prev`, `next`. Следуйте формату выходного файла из примера.

Решение

```
package week8;
```

```java
import mooc.EdxIO;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Objects;

public class Week_8_2 {

    private static Node tail;

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

        int n = edxIO.nextInt();
        Map<String, Node> map = new HashMap<>();
        for (int i = 0; i < n; i++) {
            switch (edxIO.next()) {
                case "get":
                    Node node = map.get(edxIO.next());
                    if (node == null) {
                        edxIO.println("<none>");
                    } else {
                        edxIO.println(node.getValue());
                    }
                    break;
                case "prev":
                    node = map.get(edxIO.next());
                    if (node != null && node.getPrev() != null) {
                        edxIO.println(node.getPrev().getValue());
                    } else {
                        edxIO.println("<none>");
                    }
                    break;
                case "next":
                    node = map.get(edxIO.next());
                    if (node != null && node.getNext() != null) {
                        edxIO.println(node.getNext().getValue());
                    } else {
                        edxIO.println("<none>");
                    }
                    break;
                case "put":
                    String key = edxIO.next();
                    String value = edxIO.next();
                    node = map.get(key);
                    if (node == null) {
                        node = new Node(key, value);
                        node.setPrev(tail);
                        if (tail != null) {
                            tail.setNext(node);
                        }
                        tail = node;
                        map.put(key, node);
                    } else {
                        node.setValue(value);
                    }
```

```
                break;
            case "delete":
                key = edxIO.next();
                node = map.get(key);
                if (node != null) {
                    map.remove(key);
                    Node prev = node.getPrev();
                    Node next = node.getNext();
                    if (prev != null) {
                        prev.setNext(next);
                    }
                    if (next != null) {
                        next.setPrev(prev);
                    } else if (node.equals(tail)) {
                        tail = prev;
                    }
                }
                break;
            default:
                throw new IllegalStateException();
        }
    }

    edxIO.close();
}

static class Node {
    private Node prev;
    private Node next;
    private String key;
    private String value;

    public Node(String key, String value) {
        this.value = value;
    }

    public String getValue() {
        return value;
    }

    public void setValue(String value) {
        this.value = value;
    }

    public Node getPrev() {
        return prev;
    }

    public void setPrev(Node prev) {
        this.prev = prev;
    }

    public Node getNext() {
        return next;
    }

    public void setNext(Node next) {
        this.next = next;
```

```
    }

    public String getKey() {
        return key;
    }

    public void setKey(String key) {
        this.key = key;
    }

  }


}
```

Результаты

№ теста	Результат	Время, мс	Память	Размер входного файла	Размер выходного файла
Max		2671	237117440	23499808	10303658
1	OK	156	21618688	158	26
2	OK	109	21557248	12	8
3	OK	109	21581824	25	5
4	OK	109	21626880	25	8
5	OK	109	21594112	82	20
6	OK	109	21630976	1200	504
7	OK	109	21647360	1562	564
8	OK	109	21917696	12204	4617
9	OK	125	21925888	12058	4340
10	OK	265	42725376	960183	395964
11	OK	250	44150784	1318345	765350
12	OK	234	44294144	1420595	880052
13	OK	265	42905600	1079934	395020
14	OK	234	41648128	840022	332970
15	OK	250	43724800	1223121	889998
16	OK	265	52424704	3120970	486100
17	OK	265	52539392	3123298	486652
18	OK	250	52523008	3122193	479024
19	OK	234	41836544	900630	420456
20	OK	234	52445184	3121195	486718
21	OK	359	77246464	4199992	8
22	OK	390	77254656	4099993	8

23	OK	406	77049856	3999994	8
24	OK	375	76984320	3899995	8
25	OK	421	76890112	3799996	8
26	OK	359	76849152	3699997	8
27	OK	390	76677120	3599998	8
28	OK	375	76722176	3499999	8
29	OK	468	77647872	3400000	8
30	OK	531	77496320	3300001	8
31	OK	453	70012928	5399043	1973124
32	OK	437	68947968	4200443	1669405
33	OK	453	70942720	6099290	4429770
34	OK	687	118706176	15598672	2589784
35	OK	687	118431744	15589269	2586758
36	OK	859	133378048	15603830	2398360
37	OK	484	69189632	4499616	2110630
38	OK	718	118697984	15603381	2583188
39	OK	2515	235446272	20999992	8
40	OK	2562	235474944	20499993	8
41	OK	2531	233897984	19999994	8
42	OK	2640	233582592	19499995	8
43	OK	2453	233996288	18999996	8
44	OK	2546	233627648	18499997	8
45	OK	2671	236654592	17999998	8
46	OK	2359	194330624	17499999	8
47	OK	2343	194236416	17000000	8
48	OK	2343	193073152	16500001	8
49	OK	1218	173568000	18500008	5499986
50	OK	2359	236265472	23499808	220
51	OK	531	77508608	13500208	10303658
52	OK	984	128245760	15500008	8799944
53	OK	2359	222334976	21500008	2200000
54	OK	1125	174202880	18500008	5500000
55	OK	2453	237117440	23499808	220
56	OK	531	77545472	13500208	10300130
57	OK	1046	127680512	15500008	8799958

58	OK	2421	223272960	21500008	2200000