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 № 8

ПО ДИСЦИПЛИНЕ «Алгоритмы и структуры данных»

Выполнил: Мещеряков Владимир Алевтинович

Группа: P3218

Преподаватель: Романов Алексей Андреевич

Волчек Дмитрий Геннадьевич

Санкт-Петербург

2019 г.

Week 8

1) Множество

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

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.

Пример

input.txt	output.txt
8 A 2 A 5 A 3 ? 2 ? 4 A 2 D 2 ? 2	Y N N

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

//Удаляет k-тый элемент списка list1
auto delete_list_elem(list<long long>& list1, int k)
{
	list<long long>::iterator it = list1.begin();
	std::advance(it, k); // <-- advance итерирует переданный итератор на k позиций
	if (it != list1.end())
	{
		return list1.erase(it); // <--- Вернет итератор на k+1 элемент, перед it нет *
	}
	return it;
}

//Возвращает позицию элемента со значением value, если он есть, в обратном случае возвращает -1
int find_list_elem(list<long long> list1, long long value) {
	int counter = 0;
	int size = list1.size();
	for (long long number : list1) {
		if (number == value) {
			break;
		}
		else {
			counter++;
		}
	}
	if (counter == size) {
		return -1;
	}
	return counter;
}

int main() {
	long N;
	io >> N;
	//Создаём массив листов - закрытая адресация
	list<long long>* ht = new list<long long>[N];

	char command;
	long long element;
	long hash;
	bool is_in_list;
	for (long i = 0; i < N; i++) {
		io >> command >> element;
		hash = abs(element) % N;
		switch (command)
		{
		case 'A':
			//Если такое значение уже есть в списке, то ничего не делаем, если нет, то вставляем в
начало списка
			is_in_list = false;
			for (long long number : ht[hash]) {
				if (number == element) {
					is_in_list = true;
				}
			}
			if (!is_in_list) {
				ht[hash].push_front(element);
			}
			break;
		case 'D':
			//Если такое значение есть в списке - удаляем его
			if (!ht[hash].empty()) {
				int pos = find_list_elem(ht[hash], element);
				if (pos != -1) {
					delete_list_elem(ht[hash], pos);
				}
			}
			break;
		default:
			if (!ht[hash].empty()) {
				int pos = find_list_elem(ht[hash], element);
				if (pos != -1) {
					io << "Y\n";
					break;
				}
			}
			io << "N\n";
			break;
		}
	}

	return 0;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.796	51511296	11189636	501237
1	OK	0.000	2437120	43	9
2	OK	0.015	2445312	8	3
3	OK	0.000	2433024	51	12
4	OK	0.015	2441216	542	99

2)Прошитый ассоциативный массив

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	**3 секунды**
Ограничение по памяти:	256 мегабайт

Реализуйте прошитый ассоциативный массив.

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.

Пример

input.txt	output.txt
14 put zero a put one b put two c put three d put four e get two prev two next two delete one delete three get two prev two next two next four	c b d c a e <none>

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

//Возвращает позицию предыдущего элемента для pos, если он есть, и -1 в обратном случае
int get_prev_pos(string* pos_string, int pos) {
	pos--;
	while (pos > 0 && pos_string[pos].empty())
	{
		pos--;
	}
	if (!pos_string[pos].empty() && pos > -1) {
		return pos;
	}
	else {
		return -1;
	}
}
//Возвращет позицию следующего элемента для pos, если он есть, и -1 в обратном случае
int get_next_pos(string* pos_string, int pos, long keys_counter) {
	pos++;
	while (pos < keys_counter && pos_string[pos].empty())
	{
		pos++;
	}
	if (!pos_string[pos].empty() && pos > -1 && pos < keys_counter) {
```

```
                return pos;
        }
        else {
                return -1;
        }
}

int main() {
        long N;
        io >> N;
        //Ассоциативный массив
        map<string, string> ht;
        //Массив, хранящий для каждого ключа порядок добавления в ассоциативный массив
        map<string, long> keys_pos;
        //Массив, хранящий ключи в остортированном по добавлению порядке
        string* pos_string = new string[N];
        //Счётчик введённых ключей
        long key_counter = 0;
        int pos, prev_pos;
        string command, key, element;

        for (long i = 0; i < N; i++) {
                io >> command >> key;
                switch (command[0])
                {
                case 'p':
                        //put
                        if (command[1] == 'u') {
                                io >> element;
                                if (!ht[key].empty()) {
                                        ht[key].assign(element);
                                }
                                else {
                                        ht[key].assign(element);
                                        keys_pos[key] = key_counter;
                                        pos_string[key_counter++] = key;
                                }
                                break;
                        }
                        //prev
                        else {
                                pos = keys_pos[key];
                                prev_pos = get_prev_pos(pos_string, pos);
                                if (prev_pos != -1) {
                                        io << ht[pos_string[prev_pos]] << "\n";
                                }
                                else
                                {
                                        io << "<none>\n";
                                }
                                break;
                        }
                case 'n':
                        pos = keys_pos[key];
                        if (pos <= 0 && pos_string[pos] != key) {
                                prev_pos = -1;
                        }
                        else {
                                prev_pos = get_next_pos(pos_string, pos, key_counter);
                        }
                        if (prev_pos != -1) {
                                io << ht[pos_string[prev_pos]] << "\n";
                        }
                        else
                        {
                                io << "<none>\n";
                        }
                        break;
                case 'g':
                        element = ht[key];
                        if (!element.empty()) {
                                io << element << "\n";
                        }
                        else {
                                io << "<none>\n";
                        }
                        break;
                case 'd':
                        ht[key].assign("");
                        if (keys_pos[key] == -1 || (keys_pos[key] == 0 && pos_string[0] != key)) {

                        }
                        else {
                                pos_string[keys_pos[key]] = "";
                        }
                        keys_pos[key] = -1;
                        break;
                default:
                        break;
```

```
            }
        }
        return 0;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.453	231092224	23499808	10303658
1	OK	0.015	10952704	158	26
2	OK	0.031	10887168	12	8
3	OK	0.031	10915840	25	5
4	OK	0.031	10928128	25	8
5	OK	0.031	10924032	82	20
6	OK	0.031	10977280	1200	504

3)Почти интерактивная хеш-таблица

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	**5 секунд**
Ограничение по памяти:	256 мегабайт

В данной задаче у Вас не будет проблем ни с вводом, ни с выводом. Просто реализуйте быструю хеш-таблицу.

В этой хеш-таблице будут храниться целые числа из диапазона $[0;1015-1]$. Требуется поддерживать добавление числа x и проверку того, есть ли в таблице число x. Числа, с которыми будет работать таблица, генерируются следующим образом. Пусть имеется четыре целых числа N, X, A, B, такие что:

- $1 \le N \le 107$;
- $0 \le X < 1015$;
- $0 \le A < 103$;
- $0 \le B < 1015$.

Требуется N раз выполнить следующую последовательность операций:

- Если X содержится в таблице, то установить $A \leftarrow (A+AC) \bmod 103$, $B \leftarrow (B+BC) \bmod 1015$.

- Если X не содержится в таблице, то добавить X в таблицу и установить $A \leftarrow (A+AD) \bmod 103$, $B \leftarrow (B+BD) \bmod 1015$.
- Установить $X \leftarrow (X \cdot A+B) \bmod 1015$.

Начальные значения X, A и B, а также N, AC, BC, AD и BD даны во входном файле. Выведите значения X, A и B после окончания работы.

Формат входного файла

Во первой строке входного файла содержится четыре целых числа N, X, A, B. Во второй строке содержится еще четыре целых числа AC, BC, AD и BD, такие что $0 \le AC,AD < 103$, $0 \le BC,BD < 1015$.

Формат выходного файла

Выведите значения X, A и B после окончания работы.

Пример

input.txt	output.txt
4 0 0 0 1 1 0 0	3 1 1

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

//Хеш-функция
long get_hash(long long value, long ht_size) {
    return abs((value * 47) ^ (value * 31)) % ht_size;
}
//Возвращает true если элемент вставлен, false - если такой элемент уже был добавлен
bool insert_into_ht(long long*& ht, long long value, long ht_size) {
    //Вычисляем хеш
    long hash = get_hash(value, ht_size);
    //Пока не наткнёмся на свободную ячейку или ячейку с этим же значением двигаемся вперёд на одну
ячейку
    while (ht[hash] != -1 && ht[hash] != value) {
        //Зацикливаем массив
        if (++hash == ht_size) {
            hash = 0;
        }
    }
    if (ht[hash] == value) {
        return false;
    }
    else {
        ht[hash] = value;
        return true;
```

```
    }
}

int main() {
    long N;
    int A, Ac, Ad;
    long long X, B, Bc, Bd;

    io >> N >> X >> A >> B >> Ac >> Bc >> Ad >> Bd;
    //Создаём массив в два раза большего размера
    long long* ht = new long long[N * 2];

    //-1 - обозначение для свободной ячейки
    for (long i = 0; i < N * 2; i++) {
        ht[i] = -1;
    }

    for (long i = 0; i < N; i++) {
        if (insert_into_ht(ht, X, N * 2)) {
            A = (A + Ad) % 1000;
            B = (B + Bd) % 1000000000000000;
        }
        else {
            A = (A + Ac) % 1000;
            B = (B + Bc) % 1000000000000000;
        }
        X = (X * A + B) % 1000000000000000;
    }

    io << X << " " << A << " " << B;

    return 0;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		2.046	170565632	87	37
1	OK	0.031	10252288	18	7
2	OK	0.031	10194944	19	7
3	OK	0.078	10194944	21	7
4	OK	0.015	10186752	21	7
5	OK	0.031	10207232	21	7
6	OK	0.015	10252288	21	15
7	OK	0.015	10309632	21	7