## Задача 1. Хеш-таблица (8 баллов)

Реализуйте структуру данных типа “множество строк” на основе динамической хеш-таблицы с открытой адресацией. Хранимые строки непустые и состоят из строчных латинских букв. Начальный размер таблицы должен быть равным 8-ми. Перехеширование выполняйте при добавлении элементов в случае, когда коэффициент заполнения таблицы достигает 3/4.

Хеш-функцию строки реализуйте с помощью вычисления значения многочлена методом Горнера.

Структура данных должна поддерживать операции добавления строки в множество, удаления строки из множества и проверки принадлежности данной строки множеству.

**1\_1.** Для разрешения коллизий используйте квадратичное пробирование. i-ая проба  
g(k, i)=g(k, i-1) + i (mod m). m - степень двойки.

**1\_2.** Для разрешения коллизий используйте двойное хеширование.

Формат входных данных

Каждая строка входных данных задает одну операцию над множеством. Запись операции состоит из типа операции и следующей за ним через пробел строки, над которой проводится операция.

Тип операции – один из трех символов:  
 + означает добавление данной строки в множество;   
 - означает удаление строки из множества;   
 ? означает проверку принадлежности данной строки множеству.

При добавлении элемента в множество НЕ ГАРАНТИРУЕТСЯ, что он отсутствует в этом множестве. При удалении элемента из множества НЕ ГАРАНТИРУЕТСЯ, что он присутствует в этом множестве.

Формат выходных данных

Программа должна вывести для каждой операции одну из двух строк OK или FAIL, в зависимости от того, встречается ли данное слово в нашем множестве.

|  |  |
| --- | --- |
| stdin | stdout |
| + hello + bye ? bye  + bye - bye ? bye ? hello | OK  OK  OK  FAIL  OK  FAIL  OK |

## Задача 2. Жадность (3 балла)

**2\_1.** **Атлеты.**

В город *N* приехал цирк с командой атлетов. Они хотят удивить горожан города *N* — выстроить из своих тел башню максимальной высоты. Башня — это цепочка атлетов, первый стоит на земле, второй стоит у него на плечах, третий стоит на плечах у второго и т.д.

Каждый атлет характеризуется силой *si* (kg) и массой *mi* (kg). Сила — это максимальная масса, которую атлет способен держать у себя на плечах.

К сожалению ни один из атлетов не умеет программировать, так как всю жизнь они занимались физической подготовкой, и у них не было времени на изучение языков программирования. Помогите им, напишите программу, которая определит максимальную высоту башни, которую они могут составить.

Известно, что если атлет тяжелее, то он и сильнее:

если *mi*>*mj*, то *si* > *sj*.

Атлеты равной массы могут иметь различную силу.

Формат входных данных:

Вход содержит только пары целых чисел — массу и силу атлетов. Число атлетов 1 ≤ *n* ≤ 100000. Масса и сила являются положительными целыми числами меньше, чем 2000000.

Формат выходных данных:

Выход должен содержать натуральное число — максимальную высоту башни.

|  |  |
| --- | --- |
| in | out |
| 3 4 2 2 7 6 4 5 | 3 |

**2\_2. Заявки на переговоры.**

В большой IT-фирме есть только одна переговорная комната. Желающие посовещаться заполняют заявки с желаемым временем начала и конца. Ваша задача определить максимальное количество заявок, которое может быть удовлетворено.

Число заявок ≤ 100000.

Формат входных данных:

Вход содержит только пары целых чисел — начала и концы заявок.

Формат выходных данных:

Выход должен содержать натуральное число — максимальное число заявок.

|  |  |
| --- | --- |
| in | out |
| 1 2  2 5  7 10  2 3  6 7  4 7  3 6  3 5  1 2  4 5 | 5 |

## Задача 3. ДП (3 балла)

Дано N кубиков. Требуется определить каким количеством способов можно выстроить из этих кубиков пирамиду.

Формат входных данных:

На вход подается количество кубиков N.

Формат выходных данных:

Вывести число различных пирамид из N кубиков.

**3\_1.** **Высокая пирамида.** Каждый вышележащий слой пирамиды должен быть не больше нижележащего.  
N ≤ 200.

|  |  |
| --- | --- |
| in | out |
| 3 | 3 |
| 5 | 7 |
| 7 | 15 |

**3\_2. Широкая пирамида.** Каждый вышележащий слой пирамиды должен быть строго меньше нижележащего.  
N ≤ 300.

|  |  |
| --- | --- |
| in | out |
| 3 | 2 |
| 5 | 3 |
| 7 | 5 |

## Задача 4. Динамика на дереве (6 баллов)

Дано невзвешенное дерево. Расстоянием между двумя вершинами будем называть количество ребер в пути, соединяющем эти две вершины.

**4\_1. Самая удаленная вершина.**

Для каждой вершины определите расстояние до самой удаленной от нее вершины. Время работы должно быть O(n).

Формат входных данных:

В первой строке записано количество вершин n ≤ 10000. Затем следует n – 1 строка, описывающая ребра дерева. Каждое ребро – это два различных целых числа – индексы вершин в диапазоне  
[0, n–1]. Индекс корня – 0. В каждом ребре родительской вершиной является та, чей номер меньше.

Формат выходных данных:

Выход должен содержать n строк. В i-ой строке выводится расстояние от i-ой вершины до самой удаленной от нее.

|  |  |
| --- | --- |
| in | out |
| 3  0 1  1 2 | 2  1  2 |

**4\_2. Сумма длин до остальных.**

Для каждой вершины определите сумму расстояний до всех остальных вершин. Время работы должно быть O(n).

Формат входных данных:

В первой строке записано количество вершин n ≤ 10000. Затем следует n – 1 строка, описывающая ребра дерева. Каждое ребро – это два различных целых числа – индексы вершин в диапазоне  
[0, n–1]. Индекс корня – 0. В каждом ребре родительской вершиной является та, чей номер меньше.

Формат выходных данных:

Выход должен содержать n строк. В i-ой строке выводится сумма расстояний от i-ой вершины до всех остальных.

|  |  |
| --- | --- |
| in | out |
| 3  0 1  1 2 | 3  2  3 |