**Сдать задание нужно до 18:00 1 декабря 2023 г. включительно.**

Ссылка на контест: <https://contest.yandex.ru/contest/55276/enter>

[Ведомость WEB](https://docs.google.com/spreadsheets/d/1Ak7DV9crEjXPgXODZH2m3ea38uMHujjUv2vvu_UvvF4/edit?usp=sharing)

[Ссылка на правила](https://docs.google.com/document/d/1TJHGNKRv7cbzXnZLLCawPjYl1Q71r4avJfR94AVkI-o/edit?usp=sharing)

**Общие требования для всех задач**

Ввод/вывод отделены от решения.

Не должно быть утечек памяти.

**Задача 1. Хеш-таблица (6 баллов)**

Обязательная задача

Реализуйте структуру данных типа “множество строк” на основе динамической хеш-таблицы с **открытой адресацией**. Хранимые строки непустые и состоят из строчных латинских букв.

Хеш-функция строки должна быть реализована с помощью вычисления значения многочлена методом Горнера.

Начальный размер таблицы должен быть равным 8-ми. Перехеширование выполняйте при добавлении элементов в случае, когда коэффициент заполнения таблицы достигает 3/4.

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.

**Требования:** В таблице запрещено хранение указателей на описатель элемента.

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. Порядок обхода (4 балла)**

Обязательная задача

Дано число N < 106 и последовательность целых чисел из [-231..231] длиной N.

Требуется построить бинарное дерево, заданное наивным порядком вставки.

Т.е., при добавлении очередного числа K в дерево с корнем root, если root→Key ≤ K, то узел K добавляется в правое поддерево root; иначе в левое поддерево root.

**Требования:** Рекурсия запрещена. Решение должно поддерживать передачу функции сравнения снаружи.

**2\_1. Выведите элементы в порядке in-order (слева направо).**

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

**2\_2. Выведите элементы в порядке pre-order (сверху вниз).**

| in | out |
| --- | --- |
| 3  2 1 3 | 2 1 3 |
| 3  1 2 3 | 1 2 3 |
| 3  3 1 2 | 3 1 2 |
| 4  3 1 4 2 | 3 1 2 4 |

**2\_3. Выведите элементы в порядке post-order (снизу вверх).**

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

**2\_4. Выведите элементы в порядке level-order (по слоям, “в ширину”).**

| in | out |
| --- | --- |
| 3  2 1 3 | 2 1 3 |
| 3  1 2 3 | 1 2 3 |
| 3  3 1 2 | 3 1 2 |
| 4  3 1 4 2 | 3 1 4 2 |

**Задача 3. B-дерево (4 балла)**

Постройте B-дерево минимального порядка t и выведите его по слоям.

В качестве ключа используются числа, лежащие в диапазоне 0..232 -1

**Требования:**

B-дерево должно быть реализовано в виде шаблонного класса.

Решение должно поддерживать передачу функции сравнения снаружи.

Формат входных данных

Сначала вводится минимальный порядок дерева t.

Затем вводятся элементы дерева.

Формат выходных данных

Программа должна вывести B-дерево по слоям. Каждый слой на новой строке, элементы должны выводится в том порядке, в котором они лежат в узлах.

| stdin | stdout |
| --- | --- |
| 2  0 1 2 3 4 5 6 7 8 9 | 3  1 5 7  0 2 4 6 8 9 |
| 4  0 1 2 3 4 5 6 7 8 9 | 3  0 1 2 4 5 6 7 8 9 |

**Задача 4. Использование АВЛ-дерева (5 баллов)**

Обязательная задача

**Требование для всех вариантов Задачи 4**

Решение должно поддерживать передачу функции сравнения снаружи.

**4\_1. Солдаты.** В одной военной части решили построить в одну шеренгу по росту. Т.к. часть была далеко не образцовая, то солдаты часто приходили не вовремя, а то их и вовсе приходилось выгонять из шеренги за плохо начищенные сапоги. Однако солдаты в процессе прихода и ухода должны были всегда быть выстроены по росту – сначала самые высокие, а в конце – самые низкие. За расстановку солдат отвечал прапорщик, который заметил интересную особенность – все солдаты в части разного роста. Ваша задача состоит в том, чтобы помочь прапорщику правильно расставлять солдат, а именно для каждого приходящего солдата указывать, перед каким солдатом в строе он должен становится.

**Требования:** скорость выполнения команды - O(log n).

Формат входных данных.

Первая строка содержит число N – количество команд (1 ≤ N ≤ 30 000). В каждой следующей строке содержится описание команды: число 1 и X если солдат приходит в строй (X – рост солдата, натуральное число до 100 000 включительно) и число 2 и Y если солдата, стоящим в строе на месте Y надо удалить из строя. Солдаты в строе нумеруются с нуля.

Формат выходных данных.

На каждую команду 1 (добавление в строй) вы должны выводить число K – номер позиции, на которую должен встать этот солдат (все стоящие за ним двигаются назад).

| in | out |
| --- | --- |
| 5  1 100  1 200  1 50  2 1  1 150 | 0  0  2  1 |

**4\_2. Порядковые статистики.** Дано число N и N строк. Каждая строка содержит команду добавления или удаления натуральных чисел, а также запрос на получение k-ой порядковой статистики. Команда добавления числа A задается положительным числом A, команда удаления числа A задается отрицательным числом “-A”. Запрос на получение k-ой порядковой статистики задается числом k.

**Требования:** скорость выполнения запроса - O(log n).

| in | out |
| --- | --- |
| 5  40 0  10 1  4 1  -10 0  50 2 | 40  40  10  4  50 |

**Задача 5. Алгоритм сжатия данных Хаффмана  
(6 баллов и более)**

Напишите две функции для создания архива из одного файла и извлечения файла из архива.

**Требования:** коэффициент сжатия < 1.

// Метод архивирует данные из потока original

void Encode(IInputStream& original, IOutputStream& compressed);

// Метод восстанавливает оригинальные данные

void Decode(IInputStream& compressed, IOutputStream& original);

где:

typedef unsigned char byte;

#define interface struct

interface IInputStream {

// Возвращает false, если поток закончился

virtual bool Read(byte& value) = 0;

};

interface IOutputStream {

virtual void Write(byte value) = 0;

};

В архиве сохраняйте дерево Хаффмана и код Хаффмана от исходных данных.  
Дерево Хаффмана требуется хранить эффективно - не более 10 бит на каждый 8-битный символ.

В контест необходимо отправить .cpp файл содержащий функции Encode, Decode, а также включающий файл Huffman.h. Тестирующая программа выводит размер сжатого файла в процентах от исходного.

**Лучшие 3 решения с коэффициентом сжатия < 1 из каждой группы оцениваются в 10, 7 и 4 баллов соответственно.**

Пример минимального решения:

#include "Huffman.h"

static void copyStream(IInputStream& input, IOutputStream& output)

{

byte value;

while (input.Read(value))

{

output.Write(value);

}

}

void Encode(IInputStream& original, IOutputStream& compressed)

{

copyStream(original, compressed);

}

void Decode(IInputStream& compressed, IOutputStream& original)

{

copyStream(compressed, original);

}