Задача №1494. Монобильярд

Стол для монобильярда, установленный в игровом доме уездного города N, оказался очень прибыльным вложением. До того, как в городе появился небезызвестный господин Чичиков. Раз за разом он выигрывал, и хозяин, подсчитывая убытки, понимал, что дело тут нечисто. Однако уличить подлеца в жульничестве не удавалось до прибытия в город N ревизора из Петербурга.

Правила игры в монобильярд очень просты: нужно последовательно закатить в единственную лузу шары с номерами $1, 2, \ldots, N$ (именно в этом порядке). Пока господин Чичиков играл, ревизор несколько раз подходил к столу и забирал из лузы последний закатившийся туда шар. В конце концов, оказалось, что Чичиков закатил в лузу все шары, а ревизор все шары достал и обследовал. Аферист утверждал, что закатил шары в правильном порядке. Хозяин понял, что это его шанс: ревизор должен помнить, в каком порядке он доставал шары. Однако так ли легко будет доказать жульничество?

Исходные данные:

> В первой строке записано целое число N — количество бильярдных шаров ($1 \le N \le 10^5$). В следующих N строках даны номера этих шаров в том порядке, в котором ревизор забирал их из лузы.

Результат:

> Выведите слово «Cheater», если Чичиков не мог закатить все N шаров в правильном порядке. Иначе выведите «Not a proof».

Рабочий код

```cpp
#include <iostream>
#include <stack>
using namespace std;
int main() {
    int n;
    int numberBall;
    int lastBallStack = 1;
    bool cheaterStatus = false;
    stack <int> inBall;
    cin >> n;
    cin >> numberBall;
    for (int j=lastBallStack; j<numberBall; ++j) { inBall.push(j); }
    lastBallStack = numberBall;
    for (int i=1; i<n; i++) {
        std::cin >> numberBall;
        if (inBall.empty() || numberBall > inBall.top()) {
            for (int j=lastBallStack+1; j<numberBall; ++j) { inBall.push(j); }
            lastBallStack = numberBall;
        }
        else if (numberBall == inBall.top()){ inBall.pop(); }
        else { cheaterStatus = true; }
    }
    if (cheaterStatus) { cout << "Cheater"; }
    else { cout << "Not a proof"; }
}
```

Объяснение алгоритма

Алгоритм предполагает использования стека для отслеживания ожидаемых чисел. Если следующее число больше вершины стека, оно добавляет недостающие числа в стек. Если число совпадает с вершиной стека, оно удаляется. Если число меньше и не совпадает с вершиной, алгоритм определяет, что было совершено «читерство». И в конце, если обнаружено «читерство», выводится «Cheater», иначе — «Not a proof».

Задача №1067. Структура папок

Хакер Билл случайно потерял всю информацию с жесткого диска своего компьютера, и у него нет резервных копий его содержимого. Но он сожалеет не о потере самих файлов, а о потере очень понятной и удобной структуры папок, которую он создавал и сохранял в течение многих лет работы.

К счастью, у Билла есть несколько копий списков папок с его жесткого диска. С помощью этих списков он смог восстановить полные пути к некоторым папкам (например, «WINNT\SYSTEM32\CERTSRV\CERTCO 1\X86»). Он поместил их все в файл, записав каждый найденный путь в отдельную строку.

Напишите программу, которая восстановит структуру папок Билла и выведет ее в виде отформатированного дерева.

Исходные данные:

> Первая строка содержит целое число N – количество различных путей к папкам ($1 \leq N \leq 500$). Далее следуют N строк с путями к папкам. Каждый путь занимает одну строку и не содержит пробелов, в том числе, начальных и конечных. Длина каждого пути не превышает 80 символов. Каждый путь встречается в списке один раз и состоит из нескольких имен папок, разделенных обратной косой чертой («\»).
>
> Имя каждой папки состоит из 1-8 заглавных букв, цифр или специальных символов из следующего списка: восклицательный знак, решетка, знак доллара, знак процента, амперсанд, апостроф, открывающаяся и закрывающаяся скобки, знак дефиса, собаки, циркумфлекс, подчеркивание, гравис, открывающаяся и закрывающаяся фигурная скобка и тильда (`«!\#\$\%\&'()-@^_‘\{\}~»`).

Результат:

> Выведите отформатированное дерево папок. Каждое имя папки должно быть выведено в отдельной строке, перед ним должно стоять несколько пробелов, указывающих на глубину этой папки в иерархии. Подпапки должны быть перечислены в лексикографическом порядке непосредственно после их родительской папки; перед их именем должно стоять на один пробел больше, чем перед именем их родительской папки. Папки верхнего уровня выводятся без пробелов и также должны быть перечислены в лексикографическом порядке.

Рабочий код

```cpp
#include <iostream>
#include <string>
#include <sstream>
#include <map>
using namespace std;

class Directory {
private: map<string, Directory *> root;
public: Directory() = default;

    Directory *getDir(const string& name) {
        if (root.find(name) != root.end()) return root[name];
        else return createDir(name);
    }
    Directory *createDir(const string& name) {
        root[name] = new Directory();
        return root[name];
    }

    void printTree(const string& separator = "") {
        string tabs = " ";
        tabs += separator;
        map<string, Directory *> contents(root.begin(), root.end());
        for (auto it = contents.begin(); it != contents.end(); it++) {
            cout << separator << it->first << endl;
            it->second->printTree(tabs);
        }
    }
};

int main() {
    int n;
    cin >> n;
    auto *root = new Directory();
    for (int i = 0; i < n; i++) {
        Directory *currentDir = root;
        string fullPath, name;
        cin >> fullPath;
        stringstream ss(fullPath);
        while (getline(ss, name, '\\')) { currentDir = currentDir->getDir(name); }
```

```
    }
    root->printTree();
    return 0;
}
```

Объяснение алгоритма

Алгоритм использует структуру данных hash-map, которая хранит пары ключ-значение, для представления связи вложенности между директориями и воссоздания иерархии. Это позволяет создавать вложенные структуры, имитирующие файловую систему. Алгоритм работает следующим образом:

1. Считывает полный путь к директории.
2. Разделяет путь на отдельные директории по обратному слешу как по разделителю.
3. Для каждого имени директории создаёт связь в словаре (метод `getDir`), а если папка не существует, то создает новую (метод `createDir`).
4. В конце вызывает метод `printTree`, который рекурсивно выводит всю иерархию директорий с отступами вложенности.

Задача №1322. Шпион

Спецслужбы обнаружили действующего иностранного агента. Шпиона то есть. Установили наблюдение и выяснили, что каждую неделю он через Интернет посылает кому-то странные нечитаемые тексты. Чтобы выяснить, к какой информации получил доступ шпион, требуется расшифровать информацию. Сотрудники спецслужб проникли в квартиру разведчика, изучили шифрующее устройство и выяснили принцип его работы.

На вход устройства подается строка текста $S_1 = s_1 s_2 \dots s_N$. Получив её, устройство строит все циклические перестановки этой строки, то есть $S_2 = s_2 s_3 \dots s_N s_1, \dots, S_N = s_N s_1 s_2 \dots s_{N-1}$. Затем множество строк $S_1, S_2, \dots, S_N$ сортируется лексикографически по возрастанию. И в этом порядке строчки выписываются в столбец, одна под другой. Получается таблица размером $N \times N$. В какой-то строке K этой таблицы находится исходное слово. Номер этой строки вместе с последним столбцом устройство и выдает на выход.

Например, если исходное слово S_1 = abracadabra, то таблица имеет такой вид:

1. aabracadabr = S_{11}
2. abraabracad = S_8
3. abracadabra = S_1
4. acadabraabr = S_4
5. adabraabrac = S_6
6. braabracada = S_9
7. bracadabraa = S_2
8. cadabraabra = S_5
9. dabraabraca = S_7
10. raabracadab = S_{10}
11. racadabraab = S_3

И результатом работы устройства является число 3 и строка rdarcaaaabb.

Это все, что известно про шифрующее устройство. А вот дешифрующего устройства не нашли. Но поскольку заведомо известно, что декодировать информацию можно (а иначе зачем же ее передавать?), Вам предложили помочь в борьбе с хищениями секретов и придумать алгоритм для дешифровки сообщений. А заодно и реализовать дешифратор.

Исходные данные:

> В первой и второй строках находятся соответственно целое число и строка, возвращаемые шифратором. Длина строки и число не превосходят 10^5. Строка содержит лишь следующие символы: a-z, A-Z, символ подчеркивания. Других символов в строке нет. Лексикографический порядок на множестве слов задается таким порядком символов: ABCDEFGHIJKLMNOPQRSTUVWXYZ_abcdefghijklmnopqrstuvwxyz
>
> Символы здесь выписаны в порядке возрастания.

Результат:

> Выведите декодированное сообщение в единственной строке.

Рабочий код

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main() {
    int k;
    cin >> k;
    string s;
    cin >> s;
    vector<pair<char, int>> temp(s.size());
    for (int i = 0; i < s.size(); i++) {temp[i] = {s[i], i};}
    std::sort(temp.begin(), temp.end());
    k = k - 1;
    for (int i = 0; i < s.size(); i++) {
        cout << temp[k].first;
        k = temp[k].second;
    }
    return 0;
}
```

Объяснение алгоритма

Этот код выполняет сортировку символов в строке и затем переставляет их в новом порядке декодированном порядке. Используется структура данных vector для хранения пар символов и их индексов. Алгоритм сортирует эти пары по символам, затем использует второй элемент пары (индекс) для перехода к следующему символу в отсортированной последовательности, начиная с позиции $k-1$. В результате на выходе формируется новая последовательность символов, основанная на исходных позициях символов после сортировки, которая представляет собой декодированное сообщение.