27. Вывести маршрут максимальной стоимости

Ограничение времени	1 секунда
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

В левом верхнем углу прямоугольной таблицы размером $N \times M$ находится черепашка. В каждой клетке таблицы записано некоторое число. Черепашка может перемещаться вправо или вниз, при этом маршрут черепашки заканчивается в правом нижнем углу таблицы. Подсчитаем сумму чисел, записанных в клетках, через которую проползла черепашка (включая начальную и конечную клетку). Найдите наибольшее возможное значение этой суммы и маршрут, на котором достигается эта сумма.

Формат ввода

В первой строке входных данных записаны два натуральных числа N и M, не превосходящих 100 — размеры таблицы. Далее идет N строк, каждая из которых содержит M чисел, разделенных пробелами — описание таблицы. Все числа в клетках таблицы целые и могут принимать значения от 0 до 100.

Формат вывода

Первая строка выходных данных содержит максимальную возможную сумму, вторая — маршрут, на котором достигается эта сумма. Маршрут выводится в виде последовательности, которая должна содержать N-1 букву D, означающую передвижение вниз и M-1 букву R, означающую передвижение направо. Если таких последовательностей несколько, необходимо вывести ровно одну (любую) из них.

Пример

Ввод	Вывод
5 5 9 9 9 9 9 3 0 0 0 0 9 9 9 9 9 6 6 6 6 8 9 9 9 9 9	74 D D R R R R D D

Язык GNU GCC 12.2 C++20

Набрать здесь | Отправить файл

```cpp
#include <iostream>
#include <vector>

int main() {
    int n,m;
    std::cin >> n >> m;
    std::vector<std::vector<int>> dp(n);
    std::vector<std::vector<int>> in(n);
    for (int i = 0; i != n; ++i) {
        dp[i].resize(m, -1);
        in[i].resize(m, -1);
    }
    for (int i = 0; i != n; ++i) {
        for (int j = 0; j != m; ++j) {
            std::cin >> in[i][j];
        }
    }
    dp[0][0] = in[0][0];
    for (int i = 0; i!= n; ++i) {
        for (int j = 0; j != m; ++j) {
            if (i > 0) {
                dp[i][j] = std::max(dp[i][j], dp[i - 1][j] + in[i][j]);
            }
            if (j > 0) {
                dp[i][j] = std::max(dp[i][j], dp[i][j - 1] + in[i][j]);
            }
        }
    }
    std::cout << dp[n - 1][m - 1] << "\n";
    std::vector<char> s;
    for (int i = n - 1, j = m -1; i != 0 || j != 0; ){
        if ( i > 0 && dp[i - 1][j] == dp[i][j] - in[i][j]) {
```

```
33              i--;
34              s.push_back('D');
35          } else {
36              j--;
37              s.push_back('R');
38          }
```

Отправить

Предыдущая Следующая