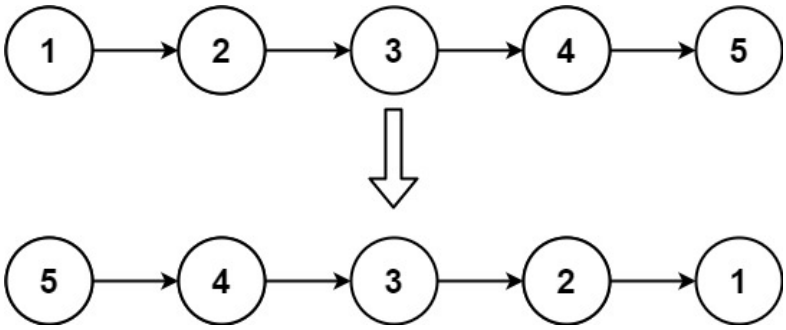
ВШЭ АиСД 2021. Списки (noSTL)

4 дек 2021, 18:54:13
старт: 8 окт 2021, 12:00:00
финиш: 17 окт 2021, 23:59:59
длительность: 9д. 11ч.
начало: 8 окт 2021, 12:00:00
конец: 17 окт 2021, 23:59:59

B. Переворот (0.15)

Ограничение времени	1 секунда
Ограничение памяти	64Mb
Ввод	стандартный ввод
Вывод	стандартный вывод

В данной задаче необходимо **развернуть** односвязный список.
Решение должно быть оптимально по памяти: объем дополнительной памяти помимо нового списка не должен никак зависить от количества узлов в связном списке.



Формат ввода

Сначала на вход подается целое число N ($0 \le N \le 2 \cdot 10^5$) - количество элементов в связном списке.
Далее на вход подаются N целых чисел ($\in [-10^9; 10^9]$) - элементов связного списка, записанные в одну строку через пробел.

Формат вывода

Элементы связного списка в обратном порядке, записанные в одну строку через пробел.

Пример 1

Ввод	Вывод
5 1 2 3 4 5	5 4 3 2 1

Пример 2

Ввод	Вывод
7 10 4 3 3 5 1 2	2 1 5 3 3 4 10

Примечания

Переворот должен быть реализован **в виде отдельной функции**.
В программе **должен быть получен** новый список, **не содержащий узлы исходного**, только копии его элементов.

Использование массивов **запрещено**.

Исходный список – **односвязный**.

Набрать здесь | Отправить файл

```
#include <iostream>

struct Node {
    int64_t value;
    Node *nextElement;

    explicit Node(int64_t value_input) {
        value = value_input;
        nextElement = nullptr;
    }
};

struct List {
    Node *headElement;
    Node *lastElement;

    List() {
        headElement = nullptr;
        lastElement = nullptr;
    }

    void pushBack(int64_t element) {
        Node *new_node = new Node(element);

        if (headElement == nullptr) {
            headElement = new_node;
            lastElement = new_node;
        } else {
            lastElement->nextElement = new_node;
            lastElement = new_node;
        }
    }

    void reverse() {
        Node *current_node = headElement;
        Node *previous_node = nullptr;
        Node *next_node;

```

Отправить

Предыдущая | Следующая