

Лучший способ в чем-то разобраться до конца — это попробовать научить этому компьютер

Дональд Кнут

Алгоритмы и структуры данных–1

2024–2025 учебный год

SET 1. Домашняя работа

Инвариант цикла. Асимптотический анализ алгоритмов.
Линейный контейнер

сентябрь							
пн	вт	ср	чт	пт	сб	вс	
2	3	4	5	6	7	8	
9	10	11	12	13	14	15	
16	17	18	19	20	21	22	
23	24	25	26	27	28	29	
30							

Немного инструкций

Задания в рамках домашней работы SET 1 подразделяются на два блока:

1. *Блок А* «Аналитические задания» — задачи, связанные с разработкой, анализом корректности и временной сложности алгоритмов.

Решения заданий Блока А оформляются в письменном виде в любом удобном формате (*LATEX*, текстовый документ, скан, изображение и др.) и загружаются для оценки в соответствующую форму раздела **SET 1. Домашняя работа** на странице курса в LMS.

2. *Блок Р* «Задания на разработку» — задачи, связанные с реализацией, использованием и обработкой линейных контейнеров.

Решения заданий Блока Р загружаются в систему **CODEFORCES** и проходят автоматизированное тестирование. Для загрузки нужно перейти на <https://dsahse.contest.codeforces.com> и выбрать соответствующее соревнование. Доступ к соревнованию предоставлен по тем же учетным данным, что и к системе Яндекс.Контест.

Домашняя работа SET 1 содержит 4 обязательные задачи в Блоке А и 5 обязательных задач в блоке Р. Баллы, которые можно получить за решение задач, распределены следующим образом:

Блок А					Блок Р				
A1	A2b	A3	A4	A5	P1	P2	P3	P4	P5
6	8	7	10	6	15	8	7	8	7

Задачи, помеченные “b” не являются обязательными — баллы за их решение относятся к *бонусным*. Подтверждение решений бонусных задач сопровождается обязательной *устной защитой*.

Важные даты

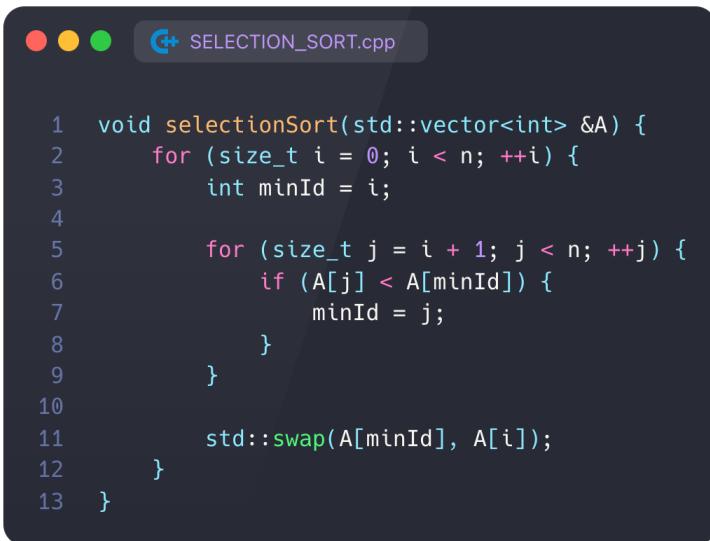
1. Домашняя работа SET 1 открыта с **14:30 16 сентября 2024 г.**
2. Прием решений на оценку завершается в **00:00 30 сентября 2024 г.**
3. Загруженное решение бонусной задачи A2b должно быть защищено до **11 октября 2024 г.**

Содержание

Задание A1.	Анализ корректности SELECTION SORT	1
Задание A2b.	Анализ корректности FAST EXPONENT	2
Задание A3.	Точная функция $T(n)$ и порядок ее роста	3
Задание A4.	Разные алгоритмы решения одной* задачи	4
Задание A5.	Поиск значения в отсортированной матрице	5
Задание P1.	Клинок, рассекающий двусвязность	6
Задание P2.	Водяное колесо!	9
Задание P3.	Распаковка Иноске	11
Задание P4.	Сортировка вагонов	13
Задание P5.	Сражение в замке бесконечности	15

Задача А1. Анализ корректности SELECTION SORT

Дана следующая итерационная реализация алгоритма сортировки выбором заданного целочисленного массива A , количество элементов в котором обозначено n :



```
1 void selectionSort(std::vector<int> &A) {
2     for (size_t i = 0; i < n; ++i) {
3         int minId = i;
4
5         for (size_t j = i + 1; j < n; ++j) {
6             if (A[j] < A[minId]) {
7                 minId = j;
8             }
9         }
10        std::swap(A[minId], A[i]);
11    }
12 }
```

1. 1.5 балла

Сформулируйте условие P_1 , которое подходит в качестве инварианта внутреннего цикла алгоритма по j . Представьте краткое обоснование (например, с использованием частичной трассировки выполнения цикла).

2. 1.5 балла

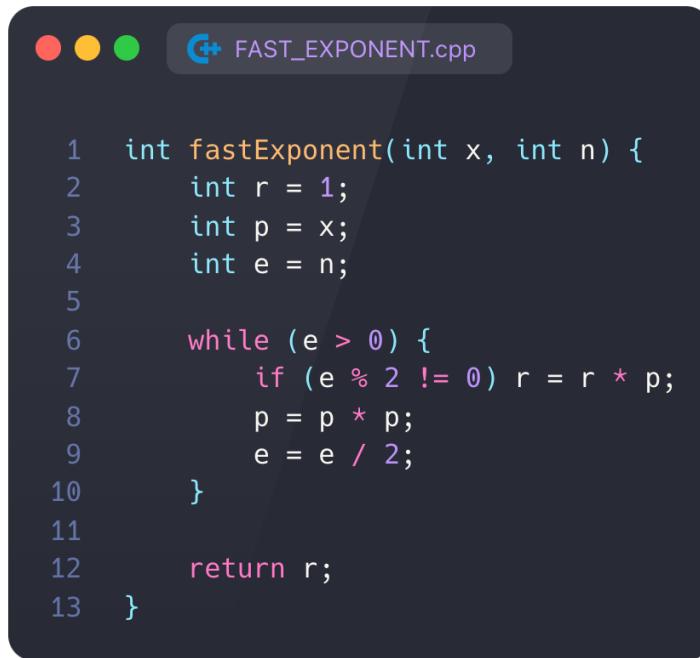
Сформулируйте условие P_2 , которое подходит в качестве инварианта внешнего цикла алгоритма по i . Представить краткое обоснование.

3. 3 балла

Выполните проверку выполнения найденных инвариантов P_1 и P_2 до входа в каждый из циклов (INIT), в каждой итерации циклов (MNT), при выходе из цикла (TRM).

Задача A2b. Анализ корректности FAST EXPONENT

Ниже приведена циклическая реализация алгоритма возведения некоторого положительного числа x в положительную степень n .



```
1 int fastExponent(int x, int n) {
2     int r = 1;
3     int p = x;
4     int e = n;
5
6     while (e > 0) {
7         if (e % 2 != 0) r = r * p;
8         p = p * p;
9         e = e / 2;
10    }
11
12    return r;
13 }
```

1. 2 балла

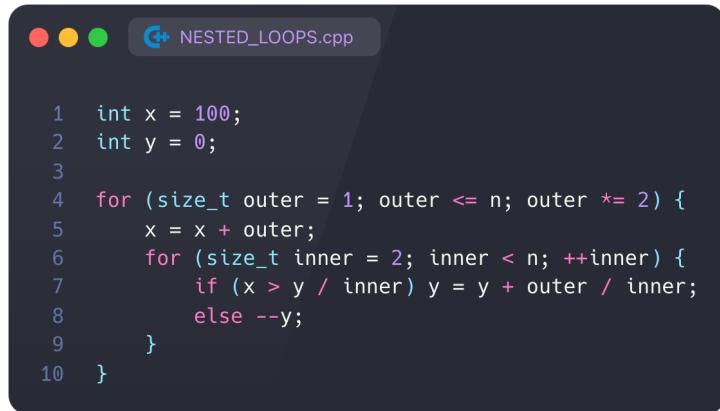
Какое точное количество операций умножения требуется выполнить, чтобы вычислить x^n с помощью алгоритма FAST EXPONENT? Всегда ли данный алгоритм лучше наивного способа вычисления?

2. 6 баллов

Сформулируйте условие P , которое подходит в качестве инварианта цикла `while`. Представить достаточное обоснование выбора инварианта. Выполнить проверку выполнения найденного инварианта P до входа в цикл (INIT), в каждой итерации цикла (MNT), а также при выходе из цикла (TRM)

Задача А3. Точная функция $T(n)$ и порядок ее роста

Дан следующий фрагмент программного кода на языке C++, где n – это некоторая заранее заданная положительная целочисленная константа:



```
1 int x = 100;
2 int y = 0;
3
4 for (size_t outer = 1; outer <= n; outer *= 2) {
5     x = x + outer;
6     for (size_t inner = 2; inner < n; ++inner) {
7         if (x > y / inner) y = y + outer / inner;
8         else --y;
9     }
10 }
```

1. 5 баллов

Составьте *точное* выражение для функции временной сложности $T(n)$ с учетом того, что арифметическая операция, присваивание и сравнение считаются *одной* элементарной операцией (каждая). В ответе представьте ход вычислений.

При составлении выражения для $T(n)$ обратите особое внимание на условный оператор — количество элементарных операций в разных ветвях условия *отличается*.

2. 2 балла

Найдите функцию $f(n)$, для которой справедливо $T(n) = \Theta(f(n))$. Обоснуйте свой ответ в соответствии с определением Θ -нотации.

Задача А4. Разные алгоритмы решения одной* задачи

Даны три алгоритма, в рамках которых выполняется обработка целочисленного массива A , содержащего n элементов, а $\text{sort}(A)$ соответствует сортировке массива некоторым способом:

```
1 algorithm1:
2     c = 0
3     ind = -1
4
5     for i = 0 to n
6         c1 = 0
7         for j = 0 to n
8             if A[i] = A[j]
9                 c1 = c1 + 1
10            if c1 > c
11                c = c1
12                ind = i
13
14    if c > n / 2 return A[ind]
```

```
1 algorithm2:
2     c = 1, ind = 0
3
4     for i = 1 to n
5         if A[ind] = A[i]
6             c = c + 1
7         else
8             c = c - 1
9
10    if c = 0
11        ind = i
12        c = 1
13
14    return A[ind]
```

```
1 algorithm3:
2     if n = 1
3         return A[0]
4
5     c = 1
6     sort(A)
7
8     for i = 1 to n
9         if A[i - 1] = A[i]
10            c = c + 1
11        else
12            if c > n / 2
13                return A[i - 1]
14            c = 1
```

1. 4 балла

Утверждается, что представленные алгоритмы должны решать одну и ту же задачу, т. е., выдавать один и тот же ответ на одинаковых входных данных.

- Согласны ли вы с этим утверждением? Результаты работы каких алгоритмов из представленных могут отличаться? Какую задачу решает каждый алгоритм?
- Приведите примеры входных данных, при которых результаты работы алгоритмов могут отличаться, а также совпадать. Поясните свой ответ с помощью трассировки (частичной) работы алгоритмов.

2. 2 балла

Вычислите асимптотическую верхнюю границу $O(f(n))$ временной сложности для каждого алгоритма. Обоснуйте свой ответ. Представлять полный расчет точного выражения функции временной сложности $T(n)$ не нужно.

3. 3 балла

Какие алгоритмы и каким образом необходимо доработать, чтобы в результате все представленные алгоритмы решали одну и ту же задачу? В ответе представьте инструкции, которые необходимо добавить или удалить.

Изменять представленные алгоритмы полностью не предполагается.

4. 1 балл

Докажите, что представленные вами доработки не ухудшают ранее вычисленные в п. 2 асимптотические верхние границы временной сложности. Представьте расчеты асимптотических верхних границ сложности доработанных алгоритмов.

Задача А5. Поиск значения в отсортированной матрице

Дана квадратная матрица A размера $N \times N$, заполненная целыми числами, а также целое число key . Матрица A характеризуется тем, что:

- значения в строках отсортированы по возрастанию, а
- значения в столбцах отсортированы по убыванию.

Предполагается, что заданная матрица заполнена *уникальными* значениям. Например, таким условиям отвечает следующая квадратная матрица:

$$\begin{pmatrix} 11 & 12 & 18 \\ 8 & 9 & 10 \\ 5 & 6 & 7 \end{pmatrix}$$

1. 4 балла

Разработайте линейный по времени алгоритм поиска значения key в заданной матрице A . Представьте описание алгоритма любым удобным способом — псевдокод, блок-схема, программный код на C++ и пр.

2. 2 балла

Выполните анализ временной сложности разработанного алгоритма, составив точное выражение функции временной сложности $T(N)$. Докажите, что $T(N) = O(N)$ в соответствии с определением асимптотической верхней границы.

Задача Р1. Клинок, рассекающий двусвязность

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Танджиро Камадо — известный мечник, первым пробудивший свою метку охотника. Однако не только талант, но и упорство влияют на способности. Поэтому сегодня наш известный мечник, находясь на тренировке столпов, получил задание не столько на смекалку, сколько на терпеливую реализацию. В какой-то момент Танджиро не выдержал такого издевательства и поручил эту задачу вам. Сможете ли вы её решить?



Требуется реализовать структуру данных — двусвязный список `List` для хранения чисел `int`. Публичный интерфейс реализуемого класса включает следующие **конструкторы**:

1. Конструктор по умолчанию для создания пустого двусвязного списка
2. Конструктор копирования для создания одного списка на основе другого
3. Конструктор, который принимает `std::vector` и на его основе создает двусвязный список

Публичный интерфейс реализуемого класса включает следующие **поля**:

1. Указатель на голову списка `head`
2. Указатель на хвост списка `tail`
3. Длина списка `_size`

Публичный интерфейс реализуемого класса включает следующие **методы вставки**:

1. Метод `push_back(int data)` для вставки нового элемента в конец списка (за `tail`)
2. Метод `push_front(int data)` для вставки нового элемента в начало списка (перед `head`)

3. Метод `insert(Node* prev, int data)` для вставки нового элемента в некоторую позицию списка после `prev`.

Если позиция `prev` является некорректной (`nullptr` или «не находится в данном списке»), то необходимо выдать исключение `std::runtime_error("Incorrect position!")`.

Публичный интерфейс реализуемого класса включает следующие **методы удаления**:

1. Метод `pop_front()` для удаления элемента из начала списка.

Если удаление невозможно, то необходимо выдать исключение `std::runtime_error("Deleting in empty list")`.

2. Метод `pop_back()` для удаления элемента из конца списка.

Если удаление невозможно, то необходимо выдать исключение `std::runtime_error("Deleting in empty list")`.

3. Метод `erase(Node* pos)` для удаления элемента, находящегося на позиции `pos`.

Если позиция `pos` является некорректной (`nullptr`), то необходимо выдать исключение `std::runtime_error("Incorrect position!")`

Публичный интерфейс реализуемого класса включает следующие общие **методы доступа** к объектам в списке:

1. Метод `front()` для доступа к значению первого элемента.

2. Метод `back()` для доступа к значению последнего элемента.

3. Метод `check_cycle()` для возврата `bool` (наличия цикла в двусвязном списке). Циклом называется такая последовательность переходов с одного элемента на следующий, что она приводит к элементу, который уже был посещён.

4. Метод `size()` для доступа к размеру списка.

5. Метод `empty()` для возврата `bool` (проверка списка на пустоту).

Публичный интерфейс реализуемого класса включает следующие общие **методы обработки** контейнера:

1. Метод `clear()` для полного удаления списка и его содержимого

2. Метод `reverse()` для обращения порядка следования элементов в списке.

Например, в результате разворота списка

$1 < - > 2 < - > 3 < - > 4 < - > 5$

должен получиться список

$5 < - > 4 < - > 3 < - > 2 < - > 1$

При выполнении разворота списка не должно создаваться промежуточных списков, т.е. обработка выполняется «по месту».

3. Метод `remove_duplicates()` для удаления дублирующихся элементов в списке.

Например, после удаления дубликатов из списка

$1 < - > 2 < - > 1 < - > 1 < - > 3$

получим список

$1 < - > 2 < - > 3$

Временная сложность удаления дубликатов должна быть ограничена $O(n)$, где n — это текущая длина списка. Встроенные методы сортировки использовать не разрешается.

4. Метод `replace(int el, int new)` для замены всех вхождений элемента `el` на `new`.

Если элемент `el` не входит в данный список, то он остается неизменным.

5. Метод `copy(const List& other)` для копирования списка `other` в текущий (в том же порядке, что и элементы `other`).

Обратите внимание, что информация в текущем списке подлежит замене, в то время как в `other` она должна остаться неизменной. Будьте внимательны с утечками памяти.

6. Метод `merge(const List& other)` для копирования списка `other` в конец текущего (в том же порядке, что и элементы `other`).

Обратите внимание, что интерфейс класса `List` дополнительно расширять не разрешается. Реализация конструкторов класса `Node` (узла списка) приведена в файле `list.h` (ссылка для скачивания находится в примечаниях).

Формат входных данных

Вам нужно загрузить определение класса `List`. Загрузка неполного определения/реализации класса не гарантирует полное прохождение тестов в рамках группы. Файл с шаблоном всех функций `release.cpp` можно также найти в примечаниях по ссылке.

Формат выходных данных

Это задача-грейдер. Вашей целью будет загрузить реализацию функций класса `List`, шаблоны всех функций даны в `release.cpp`.

Не требуется загружать что-то иное, кроме реализации предоставленных функций класса. Например, вы не можете добавить свои поля, как-то модифицировать файл `list.h`. Тестирующийся код будет с исходной версией этого файла, которая вам предоставлена.

Система оценки

Подзадача	Баллы	Дополнительные ограничения	Необходимые подзадачи	Информация о проверке
1	1	Тесты конструкторов	–	первая ошибка
2	1	Тесты вставки	1	первая ошибка
3	1.5	Тесты удаления	1-2	первая ошибка
4	1	Тесты доступа	1-3	первая ошибка
5	1.5	Тесты <code>clear()</code>	1-4	первая ошибка
6	1.5	Тесты <code>reverse()</code>	1-5	первая ошибка
7	1.5	Тесты <code>remove_duplicates()</code>	1-6	первая ошибка
8	1	Тесты <code>replace()</code>	1-7	первая ошибка
9	2	Тесты <code>copy()</code>	1-8	первая ошибка
10	2	Тесты <code>merge()</code>	1-9	первая ошибка
11	1	Тесты <code>check_cycle()</code>	1-10	первая ошибка

Замечание

Используйте предложенный шаблон кода, который доступен по ссылке:
<https://disk.yandex.ru/d/XG14Z9Td03duIw/>.

Не разрешается добавлять свои приватные поля и функции в реализацию класса, это приведёт к ошибке компиляции.

Задача Р2. Водяное колесо!

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

После предыдущей задачи Танджиро было уже всё нипочём.

Он решил отработать очередной приём дыхания воды: водяное колесо. Для улучшения техники он выписал n чисел a_i — все положения Танджиро в воздухе во время приёма.

Он понял, что можно несколько модифицировать приём, если циклически сдвинуть вправо (по часовой стрелке) положения в воздухе. Теперь Камадо хочет узнать количество таких уникальных модификаций (вращений массива), которые интереснее всего опробовать первыми?



Самыми интересными считаются модификации (циклические сдвиги) с максимальным значением Хэмминга (определение см. ниже) между исходным массивом a и результатом вращения (циклического сдвига).

Расстоянием Хэмминга d_H между двумя массивами назовем количество позиций, в которых элементы массивов отличаются друг от друга.

Например, расстояние Хэмминга $d_H(A, B)$ между двумя массивами

$$A = [1, 3, 1, 4, 5]$$

$$B = [1, 4, 5, 1, 3]$$

составляет 4, поскольку значения их элементов различаются во всех позициях, кроме первой. Нетрудно заметить, что количество возможных вращений массива длины n составляет n .

Формат входных данных

В первой строке — число n ($1 \leq n \leq 2000$), которое определяет размер массива. Во второй строке через пробел заданы n элементов массива. Гарантируется, что каждый из них по модулю не превышает 10^9 .

Формат выходных данных

Необходимо вывести количество уникальных вращений массива, дающих максимальное расстояние Хэмминга.

Система оценки

Подзадача	Баллы	Дополнительные ограничения	Необходимые подзадачи	Информация о проверке
0	—	тесты из условия	—	полная
1	1	$n \leq 10$	0	первая ошибка
2	2	$n \leq 100$	1	первая ошибка
3	3	$n \leq 1000$	1-2	первая ошибка
4	2	полные ограничения	1-3	первая ошибка

Пример

стандартный ввод	стандартный вывод
4 1 1 8 0	1

Задача Р3. Распаковка Иноске

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Иноске Хашбира известен тем, что он умеет менять положение своих внутренних органов и суставов по собственному желанию. Но на этот раз он переборщил: он запаковал себя в строку! Теперь он не может сам себя распаковать. Вы, как его хороший друг, решаете помочь ему распаковаться.



Назовём строку запакованной, если выполняется следующее условие:

$s = <\text{цифра от } 0 \text{ до } 9>[\text{подстрока_s или строка } t \text{ произвольное число раз}],$

где t — также запакованная строка, а подстрока_s не содержит цифр и квадратных скобок.
Требуется разработать программу, которая «распаковывает» строку s по следующему правилу:

$<\text{цифра } x \text{ от } 0 \text{ до } 9>[\text{строка } s] \Rightarrow \text{строка } s \text{ повторяется } x \text{ раз.}$

Формат входных данных

В единственной строке задана запакованная строка s .

Формат выходных данных

Выведите результат распаковки. Гарантируется, что длина итоговой строки не превосходит 10^6 .

Система оценки

Подзадача	Баллы	Дополнительные ограничения	Необходимые подзадачи	Информация о проверке
0	—	тесты из условия	—	полная
1	2	$s_i = [a : z]; [0 : 9]; [$	0	первая ошибка
2	2	Нет вложенных скобок	0	первая ошибка
3	3	полные ограничения	0-2	первая ошибка

Примеры

стандартный ввод	стандартный вывод
1[aaaa]	aaaa
1[a3[bb]]	abbbbb
1[1[1[1[1[e]d]c]b]a]	edcba

Замечание

Пример последовательности шагов распаковки для некоторых строк:

- 1[aaaa] \rightarrow aaaa
- 1[a3[bb]] \rightarrow 1[abbbbb] \rightarrow abbbbb

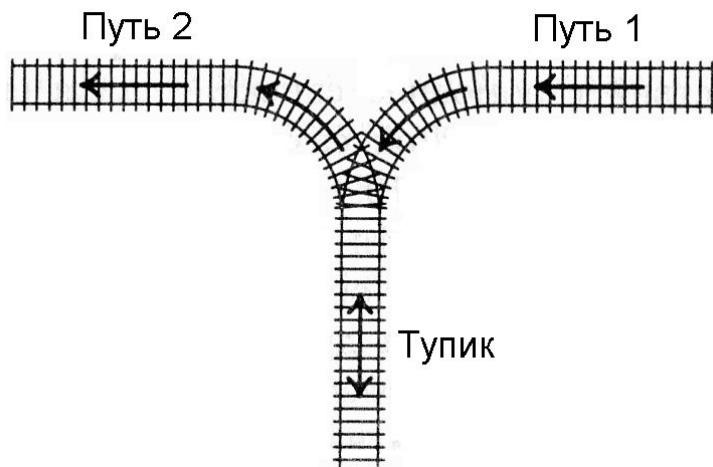
Задача Р4. Сортировка вагонов

Имя входного файла: стандартный ввод
Имя выходного файла: стандартный вывод
Ограничение по времени: 1 секунда
Ограничение по памяти: 64 мегабайта

Новая сюжетная арка: бесконечный поезд. После сражения с Эмму вам нужно решить, что делать с поездом. Вы решили, что для начала отсортируете его вагоны.



На данный момент поезд находится на пути 1. Разрешается отцепить от поезда один или сразу несколько первых вагонов и завезти их в тупик (при желании, можно даже завезти в тупик сразу весь поезд). После этого часть из этих вагонов вывезти в сторону пути 2. Затем можно завезти в тупик еще несколько вагонов и снова часть оказавшихся вагонов вывезти в сторону пути 2 и т.д. Каждый вагон может лишь один раз заехать с пути 1 в тупик, а затем один раз выехать из тупика на путь 2. Заезжать в тупик с пути 2 или выезжать из тупика на путь 1 запрещается. Кроме того, нельзя с пути 1 попасть на путь 2, не заезжая в тупик.



Известно, в каком порядке изначально идут вагоны поезда. Требуется с помощью указанных операций сделать так, чтобы вагоны поезда шли по порядку (сначала первый, потом второй и т.д., считая от головы поезда, который едет по пути 2 в сторону от тупика).

Формат входных данных

Вводится число N — количество вагонов в поезде ($1 \leq N \leq 10^4$). Дальше идут номера вагонов в порядке от головы поезда, который едет по пути 1 в сторону тупика. Вагоны пронумерованы натуральными числами от 1 до N , каждое из которых встречается ровно один раз.

Формат выходных данных

Если сделать так, чтобы вагоны шли в порядке от 1 до N , считая от головы поезда, когда поезд поедет по пути 2 из тупика, можно, выведите действия, которые нужно проделать с поездом. Каждое действие описывается двумя числами — типом и количеством вагонов:

- если нужно завезти с пути 1 в тупик K вагонов, сначала должно быть выведено число 1, а затем — число K ($K \geq 1$),
- если нужно вывезти из тупика на путь 2 K вагонов, сначала должно быть выведено число 2, а затем — число K ($K \geq 1$).

Если возможно несколько последовательностей действий, приводящих к нужному результату, выведите любую из них.

Если выстроить вагоны по порядку невозможно, выведите одно число 0.

Система оценки

Подзадача	Баллы	Дополнительные ограничения	Необходимые подзадачи	Информация о проверке
1	—	тесты из условия	—	полная
2	1	$N \leq 100$	1	первая ошибка
3	1	$N \leq 500$	1-2	первая ошибка
4	1	$N \leq 1000$	1-3	первая ошибка
5	2	$N \leq 2000$	1-4	первая ошибка
6	3	$N \leq 10000$	1-5	первая ошибка

Примеры

стандартный ввод	стандартный вывод
4 4 3 2 1	1 4 2 4
3 1 2 3	1 1 2 1 1 1 2 1 1 1 2 1
5 5 4 1 3 2	1 3 2 1 1 2 2 4

Задача Р5. Сражение в замке бесконечности

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Настало время финальной битвы! Во время сражения с Кибуцуджи Мудзаном вы с товарищами разработали какую-то тактику, и вы её придерживались.



А именно: вы решили по очереди драться с Мудзаном. Для упрощения сражения требуется разработать приложение, которое умеет обрабатывать запросы трёх видов:

1. Охотник на демонов с номером i встаёт в конец очереди.
2. Столп с номером i встаёт в середину очереди, причем при нечётной длине очереди он встает сразу за центром (он более опытный, поэтому получает такой приоритет).
3. Первый охотник из очереди начинает сражение с Кибуцуджи. Вам нужно вывести его номер.

Формат входных данных

В первой строке входных данных записано число T ($1 \leq T \leq 10^6$) — количество запросов к программе.

Следующие T строк содержат описание запросов в формате:

1. «+ i » — Охотник на демонов с номером i ($1 \leq i \leq T$) встаёт в конец очереди.
2. «* i » — Столп с номером i ($1 \leq i \leq T$) встаёт в середину очереди.
3. «-» — Первый охотник из очереди начинает сражение с Кибуцуджи. Гарантируется, что на момент такого запроса очередь не пуста.

Формат выходных данных

Для каждого запроса типа «-» программа должна вывести номер охотника, который начинает сражение.

Порядок вывода соответствует изначальному порядку следования запросов во входных данных.

Система оценки

Подзадача	Баллы	Дополнительные ограничения	Необходимые подзадачи	Информация о проверке
0	—	тесты из условия	—	полная
1	1	$n \leq 10^3$	0	первая ошибка
2	1	$n \leq 5 \cdot 10^4$	0-1	первая ошибка
3	3	$n \leq 10^5$	0-2	первая ошибка
4	2	$n \leq 10^6$	0-3	первая ошибка

Примеры

стандартный ввод	стандартный вывод
6 + 3 + 2 - + 4 + 5 -	3 2
11 + 2 + 4 * 5 - + 3 * 7 - - - - + 9	2 5 4 7 3

Замечание

Обратите внимание, что в этой задаче необходимо считывать большое количество данных. Поэтому крайне рекомендуется отключить синхронизацию потоков ввода-вывода:

```
int main() {  
    ios::sync_with_stdio(false); // Отключить синхронизацию между iostream и stdio.  
    std::cin.tie(nullptr); // Отключить синхронизацию между std::cin и std::cout.  
    ...  
}
```