Сдать задание нужно до 23 апреля 2015г. включительно. Контест останавливается в 09:00.

Ссылка на контест: http://tp-test1.tech-mail.ru/cgi-bin/new-client?contest_id=50

Задача № 1 (2 балла)

Во всех вариантах данной задачи необходимо реализовать и использовать **сортировку вставками**.

1_1. Ящики.

На склад привезли много пустых ящиков. Все ящики пронумерованы по порядку поступления от 0. Известно, что их все можно сложить один в один (то есть так, что каждый следующий помещается в предыдущий). Один ящик можно вложить в другой, если его можно перевернуть так, что размеры одного ящика по всем осям станут строго меньше размеров другого ящика по соответствующим осям. Требуется определить, в какой последовательности они будут вложены друг в друга. Вывести номера ящиков.

in	out
3 2 3 5 1 1 1 10 4 10	1 0 2
2 5 2 1 2 3 7	0 1

1_2. Ломаная 1.

Задано N точек на плоскости. Указать (N-1)-звенную несамопересекающуюся незамкнутую ломаную, проходящую через все эти точки.

Указание: стройте ломаную в порядке возрастания x-координаты. Если имеются две точки с одинаковой x-координатой, то расположите раньше ту точку, у которой y-координата меньше.

in	out
4 0 0 1 1 1 0 0 1	0 0 0 1 1 0 1 1

1_3. Ломаная 2.

Аналогично 1.2, но ломаная должна быть замкнутая. Предполагается, что никакие три точки не лежат на одной прямой.

Указание: стройте ломаную от точки, имеющей наименьшую координату x. Если таких точек несколько, то используйте точку с наименьшей координатой y.

Точки на ломаной расположите в порядке убывания углов лучей от начальной точки до всех остальных точек.

in	out
4 0 0 1 1	0 0 0 1 1 1

1 0 0 1	1 0

1_4. Строки.

Напишите программу, печатающую набор строк в лексикографическом порядке.

Строки разделяются символом перевода строки '\n'. Если последний символ в потоке ввода '\n', считать, что после него нет пустой строки. Максимальная длина строки 255 символов. Написать свою функцию сравнения строк.

in	out
4 caba abba ab aba	ab aba abba caba

Задача № 2 (4 балла)

Решение всех задач данного раздела предполагает использование кучи.

2_1. Жадина.

Вовочка ест фрукты из бабушкиной корзины. В корзине лежат фрукты разной массы. Вовочка может поднять не более K грамм. Каждый фрукт весит не более K грамм. За раз он выбирает несколько самых тяжелых фруктов, которые может поднять одновременно, откусывает от каждого половину и кладет огрызки обратно в корзину. Если фрукт весит нечетное число грамм, он откусывает большую половину. Фрукт массы 1гр он съедает полностью.

Определить за сколько подходов Вовочка съест все фрукты в корзине.

Формат входных данных. Вначале вводится n - количество фруктов и n строк с массами фруктов. Затем K - "грузоподъемность".

Формат выходных данных. Неотрицательное число - количество подходов к корзине.

in	out
3 1 2 2 2	4
3 4 3 5 6	5
7 1 1 1 1 1 1 1 3	3

2_2. Быстрое сложение.

Для сложения чисел используется старый компьютер. Время, затрачиваемое на нахождение суммы двух чисел равно их сумме.

Таким образом для нахождения суммы чисел 1,2,3 может потребоваться разное время, в зависимости от порядка вычислений.

((1+2)+3) -> 1+2 + 3+3 = 9

((1+3)+2) -> 1+3 + 4+2 = 10

((2+3)+1) -> 2+3 + 5+1 = 11

Требуется написать программу, которая определяет минимальное время, достаточное для вычисления суммы заданного набора чисел.
Формат входных данных. Вначале вводится n - количество чисел. Затем вводится n строк - значения чисел (значение каждого числа не превосходит 10^9, сумма всех чисел не превосходит 2*10^9).
Формат выходных данных. Натуральное число - минимальное время.

in	out
5 5 2 3 4 6	45
5 3 7 6 1 9	56

2_3. Тупики.
На вокзале есть некоторое количество тупиков, куда прибывают электрички. Этот вокзал является их конечной станцией. Дано расписание движения электричек, в котором для каждой электрички указано время ее прибытия, а также время отправления в следующий рейс. Электрички в расписании упорядочены по времени прибытия. Когда электричка прибывает, ее ставят в свободный тупик с минимальным номером. При этом если электричка из какого-то тупика отправилась в момент времени X, то электричку, которая прибывает в момент времени X, в этот тупик ставить нельзя, а электричку, прибывающую в момент X+1 — можно.
В данный момент на вокзале достаточное количество тупиков для работы по расписанию.
Напишите программу, которая по данному расписанию определяет, какое минимальное количество тупиков требуется для работы вокзала.
Формат входных данных. Вначале вводится n - количество электричек в расписании. Затем вводится n строк для каждой электрички, в строке - время прибытия и время отправления. Время - натуральное число от 0 до 10^9. Строки в расписании упорядочены по времени прибытия.
Формат выходных данных. Натуральное число - минимальное количеством тупиков.
Максимальное время: 50мс, память: 5Мб.

in	out
1 10 20	1
2 10 20 20 25	2
3 10 20 20 25 21 30	2

2_4. Скользящий максимум.
Дан массив натуральных чисел A[0..n), n не превосходит 10^8. Так же задан размер некотрого окна (последовательно расположенных элементов массива) в этом массиве k, k<=n. Требуется для каждого положения окна (от 0 и до n-k) вывести значение максимума в окне. Скорость работы O(n log n), память O(k).
Формат входных данных. Вначале вводится n - количество элементов массива. Затем вводится n строк со значением каждого элемента. Затем вводится k - размер окна.
Формат выходных данных. Разделенные пробелом значения максимумов для каждого положения

окна.

in	out
3 1 2 3 2	2 3
9 0 7 3 8 4 5 10 4 6 4	8 8 8 10 10 10

Задача № 3 (3 балла)

Во всех задачах данного раздела необходимо реализовать и использовать **локальную пирамидальную сортировку** (без использования дополнительной памяти). Общее время работы алгоритма O(n log n).

3_1. Реклама.

В супермаркете решили оптимизировать показ рекламы. Известно расписание прихода и ухода покупателей (два целых числа). Каждому покупателю необходимо показать минимум 2 рекламы. Рекламу можно транслировать только в целочисленные моменты времени. Покупатель может видеть рекламу от момента прихода до момента ухода из магазина.

В каждый момент времени может показываться только одна реклама. Считается, что реклама показывается мгновенно. Если реклама показывается в момент ухода или прихода, то считается, что посетитель успел её посмотреть. Требуется определить минимальное число показов рекламы.

In	Out
5 1 10 10 12 1 10 1 10 23 24	5

3_2. Современники.

Группа людей называется современниками если был такой момент, когда они могли собраться вместе. Для этого в этот момент каждому из них должно было уже исполниться 18 лет, но ещё не исполниться 80 лет.

Дан список Жизни Великих Людей. Необходимо получить максимальное количество современников. В день 18летия человек уже может принимать участие в собраниях, а в день 80летия и в день смерти уже не может.

Замечание. Человек мог не дожить до 18-летия, либо умереть в день 18-летия. В этих случаях принимать участие в собраниях он не мог.

In	Out
3 2 5 1980 13 11 2055 1 1 1982 1 1 2030 2 1 1920 2 1 2000	3

3_3. Закраска прямой 1.

На числовой прямой окрасили *N* отрезков. Известны координаты левого и правого концов каждого отрезка (L_i и R_i). Найти длину окрашенной части числовой прямой.

In	Out
3 1 4 7 8 2 5	5

3_4. Закраска прямой 2.

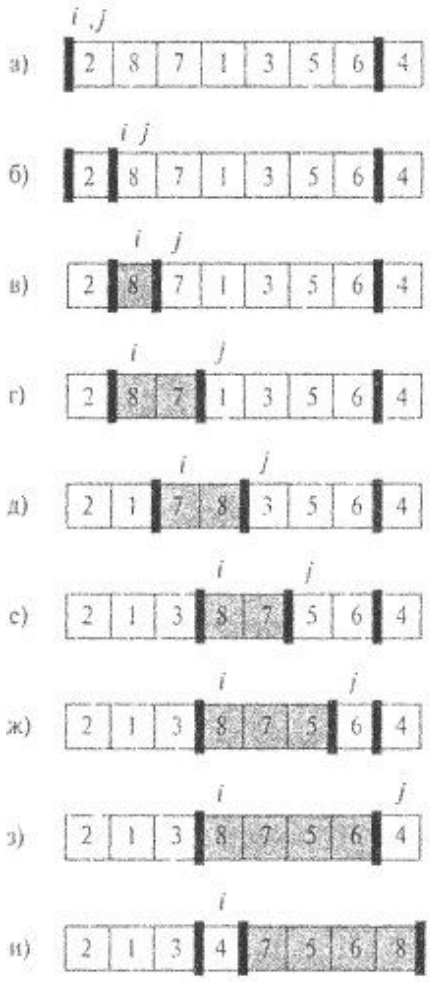
На числовой прямой окрасили *N* отрезков. Известны координаты левого и правого концов каждого отрезка (L_i и R_i). Найти сумму длин частей числовой прямой, окрашенных ровно в один слой.

In	Out
3 1 4 7 8 2 5	3

Задача № 4 (3 балла)

Даны неотрицательные целые числа n,k и массив целых чисел из [0..10^9] размера n. Требуется найти k-ю порядковую статистику. т.е. напечатать число, которое бы стояло на позиции с индексом k (0..n-1) в отсортированном массиве. Напишите нерекурсивный алгоритм.
Требования к дополнительной памяти: O(n). Требуемое среднее время работы: O(n).
Функцию Partition следует реализовывать методом прохода двумя итераторами в одном направлении. Описание для случая прохода от начала массива к концу:

- Выбирается опорный элемент. Опорный элемент меняется с последним элементом массива.
- Во время работы Partition в начале массива содержатся элементы, не бОльшие опорного. Затем располагаются элементы, строго бОльшие опорного. В конце массива лежат нерассмотренные элементы. Последним элементом лежит опорный.
- Итератор (индекс) i указывает на начало группы элементов, строго бОльших опорного.
- Итератор j больше i, итератор j указывает на первый нерассмотренный элемент.
- Шаг алгоритма. Рассматривается элемент, на который указывает j. Если он больше опорного, то сдвигаем j.
  Если он не больше опорного, то меняем a[i] и a[j] местами, сдвигаем i и сдвигаем j.
- В конце работы алгоритма меняем опорный и элемент, на который указывает итератор i.



4_1. Реализуйте стратегию выбора опорного элемента “медиана трёх”. Функцию Partition реализуйте методом прохода двумя итераторами от начала массива к концу.

4_2. Реализуйте стратегию выбора опорного элемента “медиана трёх”. Функцию Partition реализуйте методом прохода двумя итераторами от конца массива к началу.

4_3. Реализуйте стратегию выбора опорного элемента “случайный элемент”. Функцию Partition реализуйте методом прохода двумя итераторами от начала массива к концу.

4_4. Реализуйте стратегию выбора опорного элемента “случайный элемент”. Функцию Partition реализуйте методом прохода двумя итераторами от конца массива к началу.

In	Out
10 4 1 2 3 4 5 6 7 8 9 10	5

10 0 3 6 5 7 2 9 8 10 4 1	1
10 9 0 0 0 0 0 0 0 0 0 1	1

Задача № 5 (3 балла)

5_1. Первые k элементов длинной последовательности.

Дана очень длинная последовательность целых чисел длины n. Требуется вывести в отсортированном виде её первые k элементов. Последовательность может не помещаться в память. Время работы O(n * log(k)). Доп. память O(k). Использовать слияние.

In	Out
9 4 3 7 4 5 6 1 15 4 2	1 2 3 4

5_2. Сортировка почти упорядоченной последовательности.

Дана последовательность целых чисел a1...an и натуральное число k, такое что для любых i, j: если j >= i + k, то a[i] <= a[j]. Требуется отсортировать последовательность. Последовательность может быть очень длинной. Время работы O(n * log(k)). Доп. память O(k). Использовать слияние.

In	Out
10 4 0 4 3 2 1 8 7 6 5 9	0 1 2 3 4 5 6 7 8 9

5_3. Количество инверсий.

Дана последовательность целых чисел из диапазона (-10^9 .. 10^9). Длина последовательности не больше 10^6. Числа записаны по одному в строке. Количество чисел не указано.
Пусть количество элементов n, и числа записаны в массиве a = a[i]: i из [0..n-1].
Требуется напечатать количество таких пар индексов (i,j) из [0..n-1], что (i < j и a[i] > a[j]).
Указание: количество инверсий может быть больше 4*10^9 - используйте int64_t.

```
#include <stdint.h>
int64_t cnt = 0;
printf("%ld", cnt);
```

In	Out
1 2 3 4	0
4 3 2 1	6
3 2	2

2	

Задача № 6 (3 балла)

6_1. MSD для строк.

Дан массив строк. Количество строк не больше 10^5. Отсортировать массив методом поразрядной сортировки MSD по символам. Размер алфавита - 256 символов. Последний символ строки = '\0'.

In	Out
ab a aaa aa	a aa aaa ab

6_2. LSD для long long.

Дан массив неотрицательных целых 64-разрядных чисел. Количество чисел не больше 10^6. Отсортировать массив методом поразрядной сортировки LSD по байтам.

In	Out
3 4 1000000 7	4 7 1000000

6_3. Binary MSD для long long.

Дан массив неотрицательных целых 64-разрядных чисел. Количество чисел не больше 10^6. Отсортировать массив методом MSD по битам (бинарный QuickSort).

In	Out
3 4 1000000 7	4 7 1000000

Задача № 7 (6 баллов). Соревнование

7. Быстрейшая сортировка.

Задача размещениа в специальном контесте:
http://tp-test1.tech-mail.ru/cgi-bin/new-client?contest_id=52
В новом интерфейсе: http://tp-test1.tech-mail.ru:5000/contest/52/1/

Дан массив данных типа `BlackInt` размером N < 10^10.
Требуется упорядочить его по возрастанию с использованием попарных сравнений элементов.
Решение должно содержать реализацию функции

```
void sort(BlackInt *begin, BlackInt *end);
```

Разрешается использовать операции сравнения: < > <= >= == !=,
операции присвоения между элементами типа BlackInt:

```
BlackInt a = b;
std::swap(begin[0], begin[1]);
begin[0].swap(begin[1]);
```

Запрещается выполнять какие-либо операции с экземплярами класса `BlackInt` кроме перечисленных выше. В частности, преобразования типа или преобразование типа указателя на `BlackInt`.

Каждая операция сравнения стоит 1 балл.
Каждое присвоение или операция swap стоит также 1 балл.

Выигрышным объявляется решение, прошедшее тесты с минимальной стоимостью.
Решения, результаты по тестам, прошедшим решения отображаются в таблице:
http://tp-test1.tech-mail.ru:82/leaderboard/
формат: runid score cmp_count swap_copy_count runtime login datetime verdict
Файл отсортирован по убыванию стоимости решения. Файл обновляется раз в 30 секунд.
Пример решения, которе проходит тесты:

```
#include <algorithm>
void sort(BlackInt *begin, BlackInt *end) {
	std::sort(begin, end);
}
```

Перед компиляцией к решению добавляется заголовок, организующий взаимодействие с тестовой системой.

За основу должен быть взят алгоритм быстрой сортировки.
Набор оптимизаций, который необходимо реализовать:

1. Оптимизация выбора опорного элемента.
2. Оптимизация Partition.
3. Оптимизация концевой рекурсии.
4. Написать без рекурсии.