Сдать задание нужно до 28 октября 2019г. (9:00).
Контест: https://contest.yandex.ru/contest/14538/enter/
Ведомость:
https://docs.google.com/spreadsheets/d/1uzNYke_nTVsZv1fLLetIbS9lE96edpC2ofh5UuxbPFo

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

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 1 0 0 1	0 0 0 1 1 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

1_5. База данных.

В базе данных хранится N записей, вида (Name, a_1, a_2, … , a_k) — во всех записях одинаковое число параметров.

На вход задачи подаётся приоритет полей — перестановка на числах 1,...,k — записи нужно вывести в соответствии с этим приоритетом. В случае, если приоритет полей таков: 3 4 2 1, то это следует воспринимать так: надо читать как: приоритет значений из 3 колонки самый высокий, приоритет значений из колонки 4 ниже, приоритет значений из колонки 2 ещё ниже, а приоритет значений из колонки 1 самый низкий.

Входные данные:

$N\ (1 \leqslant N \leqslant 1000)$

$k\ (1 \leqslant k \leqslant 10)$

$p_1\ p_2\ ...\ p_k$ (перестановка на k числах, разделитель пробел)

N строк вида

Name $a_1\ a_2\ ...\ a_k$ (разделитель — пробел)

Выходные данные:

N строк с именами согласно приоритету

in	out
3 3 2 1 3 A 1 2 3 B 2 1 3 C 3 1 2	A C B

Задача № 2 (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).

2_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

2_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

2_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

2_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.

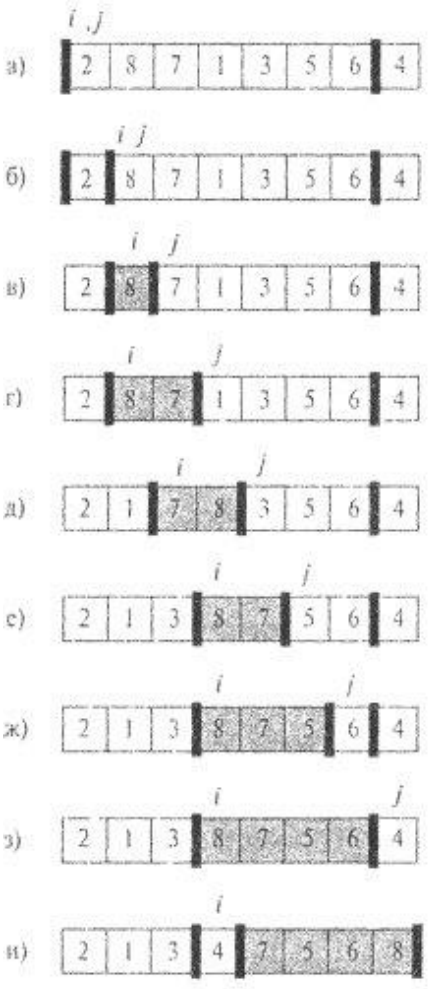
In	Out
3 1 4 7 8 2 5	3

Задача № 3 (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.



3_1. Реализуйте стратегию выбора опорного элемента “медиана трёх”. Функцию Partition реализуйте методом прохода двумя итераторами от начала массива к концу.

3_2. Реализуйте стратегию выбора опорного элемента “медиана трёх”. Функцию Partition реализуйте методом прохода двумя итераторами от конца массива к началу.

3_3. Реализуйте стратегию выбора опорного элемента “случайный элемент”. Функцию Partition реализуйте методом прохода двумя итераторами от начала массива к концу.

3_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 № 4 (4 балла)

4_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

4_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

4_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 № 5 (3 балла)

5_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

5_2. LSD для long long.

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

In	Out
3 4 1000000 7	4 7 1000000

5_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

5_4 Очень быстрая сортировка

Имеется рекуррентная последовательность A1, A2, ..., AN, строящаяся по следующему правилу:

A1 = K

Ai+1 = (Ai * M) % (2^32-1) % L

Требуется найти сумму всех нечётных по порядку элементов в отсортированной по неубыванию последовательности по модулю L.

Формат ввода: N K M L
Для входных данных
5 7 13 100
последовательность будет такой:
(7, 7*13%100=91, 91*13%100=83, 83*13%100=79, 79*13%100=27), то есть, (7, 91, 83, 79, 27)
Отсортированная последовательность
(7, 27, 79, 83, 91)
Сумма элементов на нечётных местах = (7 + 79 + 91) mod 100 = 77
Для представления элементов последовательности необходимо использовать тип данных unsigned int.
Указание
Для получения массива используйте цикл

```
a[0] = K;
for (int i = 0; i < N-1; i++)
    a[i+1] = (unsigned int)((a[i]*(unsigned long long)M)&0xFFFFFFFFU)modL;
```

In	Out
5 7 13 100	77

Задача № 6 (5 балла)

Быстрейшая сортировка. Задача в контесте не представлена.
Реализуйте сортировку, основанную на QuickSort.
Протестируйте скорость её работы на массиве из миллиона рандомных элементов.
Попробуйте оптимизировать её работу. Минимальный набор оптимизаций, который необходимо реализовать:
1. Оптимизация выбора опорного элемента
2. Оптимизация Partition
3. Написать с одной ветвью рекурсии
4. Написать без рекурсии
5. Оптимизация концевой рекурсии
Для сдачи необходимо предоставить табличку показывающую влияние оптимизаций на время выполнения. Каждый замер необходимо выполнить минимум 3 раза.

Соревнование

Сдается в отдельном контесте: https://contest.yandex.ru/contest/14852
Дедлайн на контест для соревнования УВЕЛИЧЕН до **4 ноября 2019г. (9:00)**.

Дан массив целых чисел в диапазоне [0..10^9]. Размер массива кратен 10 и ограничен сверху значением $2.5 * 10^6$ элементов. Все значения массива являются элементами псевдо-рандомной последовательности.
Нужно написать функцию сортировки массива целых чисел

```
void Sort(unsigned int* arr, unsigned int size)
```

Необходимо отсортировать элементы массива за минимальное время. Разрешается использовать любую сортировку, написанную самостоятельно. Сортировка обязательно должна сортировать все элементы массива.

Файл с решением НЕ ДОЛЖЕН содержать main, только функцию сортировки. Должен включать файл *#include “sort.h”*

Пример сортировки пузырьком удовлетворяющей условиям:

```
#include "sort.h"

void Sort(unsigned int* arr, unsigned int size) {
for (int i = 0; i < size; i++)
  for (int j = i + 1; j < size; j++) {
    if(arr[j] < arr[i]) {
      unsigned int value = std::move(arr[j]);
      arr[j] = std::move(arr[i]);
      arr[i] = std::move(value);
    }
  }
}
```

Соревнование общее на весь курс. Доп баллы получают 10 самых быстрых сортировок.
Максимальный бонус + 10 баллов выдается за самую быструю сортировку.
Одиннадцатый результат не получает бонуса.
Остальные баллы выдаются пропорционально близости к лучшему результату.