Контрольная работа №1

В рамках данной контрольной работы необходимо решить предложенные задачи на языке программирования высокого уровня из предложенного перечня. Варианты задач находятся в отдельном файле. В качестве результата выполнения практической работы необходимо приложить **архив с файлами решенных задач, а также отчет о выполнении работы**. Файлы решенных задач представляют собой файлы с исходным кодом с установленным расширением. При решении задач необходимо писать свои собственные функции и структуры данных, использование встроенных не допускается.

Отчет о выполнении практической работы должен содержать:

1. Титульный лист
2. Содержание
3. параграфы, в которых раскрыто решение каждой задачи. По каждой задаче необходимо представить следующую информацию:

3.1. *Условие задачи*. Берется из файла.

3.2. *Ход решения задачи*. Студент описывает логику решения данной задачи. Какие алгоритмы и для чего использованы, как построена программа. Данная часть является описательной. Здесь следует говорить именно о построении алгоритма, опуская процессы ввода и вывода данных (если это не является основной сутью алгоритма).

3.3. *Листинг программы с комментариями*. Копируется весь программный код.

3.4. *Тестирование программы*. Если для задачи предусмотрены автотесты, приложить скриншот их прохождения. Если нет: Составляется таблица, содержащая следующие поля: номер теста, входные данные, результат выполнения программы, корректность, время выполнения (мс), затраченная память (Мб). Составляется не менее 10-ти тестовых наборов данных согласно условию задачи. Тестовые наборы входных данных студент составляет самостоятельно. В обязательном порядке программа тестируется на граничных наборах входных данных (например, если N варьируется от 0 до 10^9, то обязательно рассмотреть решение задачи при N=0 и при N близком к 10^9). Если написанная программа не позволяет решить задачу при граничных входных данных, все равно включить в тест и в качестве результата написать "Не решено". В столбце "входные данные" данные впечатываются вручную, в столбце "результат..." представляется скриншот выполнения программы (если не влезает на одну страницу, делать несколько скриншотов).

Каждая контрольная работа защищается на занятии преподавателю.

Задачи по теме 1. Введение в алгоритмы и структуры данных. Рекурсия

Задача 1. Ближайший ноль.

Ограничение по времени: 1.6 с. Ограничение по памяти: 400 Mb.

Улица, на которой хочет жить Тимофей, имеет длину n, то есть состоит из n одинаковых идущих подряд участков. На каждом участке либо уже построен дом, либо участок пустой. Тимофей ищет место для строительства своего дома. Он очень общителен и не хочет жить далеко от других людей, живущих на этой улице.

Чтобы оптимально выбрать место для строительства, Тимофей хочет для каждого участка знать расстояние до ближайшего пустого участка. (для пустого участка эта величина будет равна нулю – расстояние до самого себя)

Ваша задача – помочь Тимофею посчитать искомые расстояния. Для этого у вас есть карта улицы. Дома в городе Тимофея нумеровались в том порядке, в котором строились, поэтому их номера на карте никак не упорядочены. Пустые участки обозначены нулями.

Формат входных данных:

В первой строке дана длина улицы – n ($1 \le n \le 10^6$). В следующей строке записаны n целых неотрицательных чисел – номера домов и обозначения пустых участков на карте (нули). Гарантируется, что в последовательности есть хотя бы один ноль. Номера домов (положительные числа) уникальны и не превосходят 10^9.

Формат выходных данных:

Для каждого из участков выведите расстояние до ближайшего нуля. Числа выводите в одну строку, разделяя их пробелами.

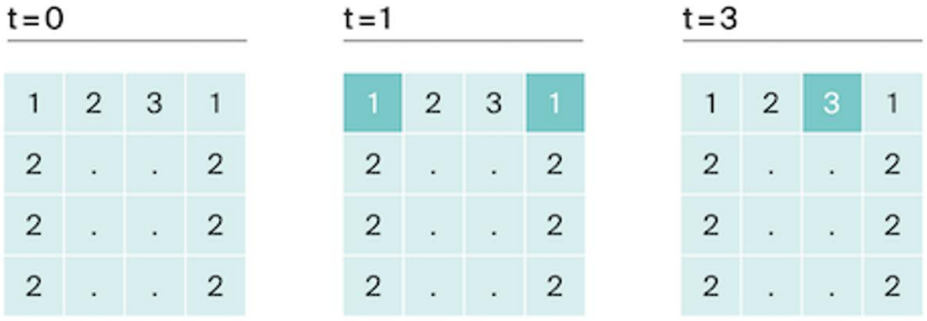
Примеры:

Стандартный ввод	Стандартный вывод
5 0 1 4 9 0	0 1 2 1 0
6 0 7 9 4 8 20	0 1 2 3 4 5

Задача 2. Ловкость рук.

Ограничение по времени: 1 с. Ограничение по памяти: 64 Mb.

Гоша и Тимофей нашли необычный тренажер для скоростной печати и хотят освоить его. Тренажер представляет собой поле из клавиш 4×4, к котором на каждом раунде появляется конфигурация цифр и точек. На клавише написана либо точка, либо цифра от 1 до 9. В момент времени t игрок дожен одновременно нажать на все клавиши, на которых написана цифра t. Гоша и Тимофей могут нажать в один момент времени на k клавиш каждый. Если в момент времени t были нажаты все нужные клавиши, то игроки получают 1 балл. Найдите число баллов, которое смогут заработать Гоша и Тимофей, если будут нажимать на клавиши вдвоём.



Формат входных данных:

В первой строке дано целое число k ($1 \leq k \leq 5$). В четырех следующих строках заданвид тренажера – по 4 символа в каждой строке. Каждый символ – либо точка, либо цифра от 1 до 9. Символы одной строки идут подряд и не разделены пробелами.

Формат выходных данных:

Выведите едиственное число – максимальное количество баллов, которое смогут набрать Гоша и Тимофей.

Примеры:

Стандартный ввод	Стандартный вывод
3 1231 2..2 2..2 2..2	2
4 1111 9999 1111 9911	1
4 1111 1111 1111 1111	0

Задача 3. Симметрическая разность.

Ограничение по времени: 2 с. Ограничение по памяти: 64 Mb.

На вход подается множество чисел в диапазоне от 1 до 20000, разделенных пробелом. Они образуют множество A. Затем идет разделитель – число 0 и на вход подается множество чисел B, разделенных пробелом, 0 – признак конца описания множества (во множество не входит). Необходимо вывести множество $A \Delta B$ – симметрическую разность множеств A и B в порядке возрастания элементов. В качестве разделителя используйте пробел. В случае, если множество пусто, вывести 0.

Формат входных данных:

1 2 3 4 5 0 1 7 5 8 0

Формат выходных данных:

2 3 4 7 8

Примеры:

Стандартный ввод	Стандартный вывод
1 2 6 8 7 3 0 4 1 6 2 3 9 0	4 7 8 9

Замечание. Для вывода можно использовать любой алгоритм сортировки.

Задача 4. Два массива.

Ограничение по времени: 2 с. Ограничение по памяти: 64 Mb.

Даны два упорядоченных по неубыванию массива. Требуется найти количество таких элементов, которые присутствуют в обоих массивах. Например, в массивах (0, 0, 1, 1, 2, 3) и (0, 1, 1, 2) имеется четыре общих элемента – (0, 1, 1, 2).

Первая строка содержит размеры массивов N1 и N2. В следующих N1 строках содержатся элементы первого массива, в следующих за ними N2 строках – элементы второго массива.

Программа должна вывести ровно одно число – количество общих элементов.

Формат входных данных:

N_a, N_b

a_1

a_2

…

a_{Na}

b_1

b_2

…

b_{Nb}

Формат выходных данных:

Одно целое число – количество общих элементов

Примеры:

Стандартный ввод	Стандартный вывод
5 5 1 1 2 2 3 0 1 3 3 4	2

Задача 5. Длинное сложение и вычитание

Ограничение по времени: 2 с. Ограничение по памяти: 64 Mb.

На вход подается три строки. Первая содержит представление длинного десятичного числа (первый операнд), вторая – представление операции, строки + и -, третья – представление второго операнда.

Длина первой и третьей строки ограничены 1000 символами. Вторая строка содержит ровно один символ.

Требуется исполнить операцию и вывести результат в десятичном представлении.

Формат входных данных:

123

+

999

Формат выходных данных:

1122

Примеры:

Стандартный ввод	Стандартный вывод
232 + -100	132
-100 - 199	-299

Замечание. Постарайтесь реализовать программу таким образом, чтобы ей можно было воспользоваться в дальнейшем. В других работах нашего курса имеются задачи, в которых потребуется длинная арифметика.

Задача 6. Вычисление полинома.

Ограничение по времени: 1 с. Ограничение по памяти: 16 Mb.

Вычисление полинома – необходимая операция для многих алгоритмов. Нужно вычислить значение полинома

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x^1 + a_0$$

Так как число n может быть достаточно велико, требуется вычислить значение полинома по модулю M. Сделать это предлагается для нескольких значений аргумента.

Формат входных данных:

Первая строка файла содержит три числа – степень полинома $2 \le N \le 100000$, количеств вычисляемых значений аргумента $1 \le M \le 10000$ и модуль $10 \le MOD \le 10^9$.

Следующие N+1 строк содержат значения коэффициентов полинома $0 \le a_i \le 10^9$

В очередных M строках содержатся значения аргументов $0 \le x_i \le 10^9$.

Формат выходных данных:

Выходной файл должен состоять из ровно M строк – значений данного полинома при заданных значениях аргументов по модулю MOD.

Примеры:

Стандартный ввод	Стандартный вывод
2 5 10 1 5 4 0 1 2 3 4	4 0 8 8 0
5 9 10 1 0 0 0	1 2 3 4 5

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

Задача 7. Две кучи.

Ограничение по времени: 2 с. Ограничение по памяти: 64 Mb.

Имеется $2 \le N \le 23$ камня с целочисленными весами W_1, W_2, … W_N. Требуется разложить их на две кучи таким образом, чтобы разница в весе куч была минимальной. Каждый камень должен принадлежать ровно одной куче.

Формат входных данных:

N

W1 W2 W3 … WN

Формат выходных данных:

Минимальная неотрицательная разница в весе куч

Примеры:

Стандартный ввод	Стандартный вывод
5 8 9 6 9 8	4
6 14 2 12 9 9 8	2

Задача 8. Магараджа.

Ограничение по времени: 1 с. Ограничение по памяти: 16 Mb.

Магараджа — это шахматная фигура, сочетающая возможности ферзя и коня. Таким образом, магараджа может ходить и бить на любое количество клеток по диагонали, горизонтали и вертикали (т.е. как ферзь), а также либо на две клетки по горизонтали и на одну по вертикали, либо на одну по горизонтали и на две по вертикали (как конь).

Ваша задача — найти число способов расставить на доске N на N ровно K магараджей так, чтобы они не били друг друга.

Формат входных данных:

Входной файл INPUT.TXT содержит два целых числа: N и K ($1 \le K \le N \le 10$).

Формат выходных данных:

В выходной файл OUTPUT.TXT выведите ответ на задачу.

Примеры:

INPUT.TXT	OUTPUT.TXT
3 1	9
4 2	20

5 3	48

Источник задачи: здесь.

Задача 9. Вырубка деревьев.

Ограничение по времени: 1 с. Ограничение по памяти: 16 Mb.

Король Флатландии решил вырубить некоторые деревья, растущие перед его дворцом. Деревья перед дворцом короля посажены в ряд, всего там растет n деревьев, расстояния между соседними деревьями одинаковы.

После вырубки перед дворцом должно остаться m деревьев, и расстояния между соседними деревьями должны быть одинаковыми. Помогите королю выяснить, сколько существует способов вырубки деревьев.

Требуется написать программу, которая по заданным числам n и m определит, сколько существует способов вырубки некоторых из n деревьев так, чтобы после вырубки осталось m деревьев и соседние деревья находились на равном расстоянии друг от друга.

Формат входных данных:

Входной файл INPUT.TXT содержит два целых числа n и m ($0 \le m , n \le 1000$).

Формат выходных данных:

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число — искомое число способов.

Примеры:

INPUT.TXT	OUTPUT.TXT
5 3	4

Примечание:

Если обозначить условно исходное расположение деревьев перед дворцом как «TTTTT», то возможные результаты после вырубки следующие:

«TTT..», «.TTT.», «..TTT», «T.T.T».

Источник задачи: здесь.

Задача 10. Перетягивание каната.

Ограничение по времени: 3 с. Ограничение по памяти: 16 Mb.

Для участия в соревнованиях по перетягиванию каната зарегистрировалось N человек. Некоторые из участников могут быть знакомы друг с другом. Причем, если двое из них имеют общего знакомого, то это не означает, что они обязательно знакомы друг с другом.

Организаторы соревнований заинтересованы в их качественном проведении. Они хотят разделить всех участников на две команды так, чтобы в первой команде было K человек, а во второй – N-K человек. Из всех возможных вариантов формирования команд, организаторы хотят выбрать такой вариант, при котором сумма сплоченностей обеих команд максимальна. Сплоченностью команды называется количество пар участников этой команды, знакомых друг с другом. Ваша задача – помочь организаторам найти требуемое разделение участников на две команды.

Формат входных данных:

В первой строке входного файла INPUT.TXT задаются три числа N, K, M, разделенные одиночными пробелами, где N – общее число зарегистрированных

участников, K – требуемое количество человек в первой команде, M – количество пар участников, знакомых друг с другом.

Каждая из следующих M строк содержит два различных числа, разделенные пробелом – номера двух участников, знакомых друг с другом. Все участники нумеруются от 1 до N.

Ограничения: все числа целые, $0 < K < N < 25$, $0 \le M \le N(N-1)/2$

Формат выходных данных:

Выходной файл OUTPUT.TXT должен содержать одну строку, состоящую из K чисел, каждое из которых задает номер участника, попавшего в первую команду. Числа должны быть разделены пробелами. Если существует несколько решений данной задачи, то выведите любое из них.

Примеры:

INPUT.TXT	OUTPUT.TXT
5 3 3 1 3 2 5 5 4	5 2 4

Источник задачи: здесь.

Задачи по теме 2. Алгоритмы сортировки

Задача 11. Поиск в сломанном массиве

Ограничение по времени: 0.001 с. Ограничение по памяти: 64 Mb.

Алла ошиблась при копировании из одной структуры данных в другую. Она хранила массив чисел в кольцевом буфере. Массив был отсортирован по возрастанию, и в нем можно было найти элемент за логарифмическое время. Алла скопировала данные из кольцевого буфера в обычный массив, но сдвинула данные исходной отсортированной последовательности. Теперь массив не является отсортированным Тем не менее, нужно обеспечить возможность находить внем элемент за O($\log n$).

Можно предполагать, что в массиве только уникальные элементы.

Задачу необходимо сдавать, выбрав компилятор Make! Решение отправляется файлом. Требуемые сигнатуры функций лежат в заготовках кода на диске.

От вас требуется реализовать функцию, осуществляющую поиск в сломанном массиве. Файлы с заготовками кода, содержащими сигнатуры Функций и базовый тест для поддерживаемых языков, находятся на Яндекс Диске по ссылке. Обратите внимание, что считывать данные и выводить ответ не требуется

Разрешение файла должно соответствовать языку на котором вы пишете (.cpp, .java, .go, .js, .py). Если вы пишете на Java, назовите файл с решением Solution.java, для C# — Solution.cs. Для остальных языков название может быть любым, кроме solution.ext где ext — разрешение для вашего языка.

Формат входных данных:

Функция принимает массив натуральных чисел и искомое число k. Длина массива не превосходит 10000. Элементы массива и число k не превосходят по значению 10000.

В примерах:

В первой строке записано число n — длина массива.

Во второй строке записано положительное число k — искомый элемент

Далее в строку через пробел записано n натуральных чисел. каждое из которых не превосходит 200000

Формат выходных данных:

Функция должна вернуть индекс элемента, равного k, если такой есть в массиве (нумерация с нуля). Если элемент не найден, функция должна вернуть — 1.

Изменять массив нельзя.

Для отсечения неэффективных решений ваша функция будет запускаться от 100000 до 1000000 раз.

Примеры:

Стандартный ввод	Стандартный вывод
2 5 19 21 100 101 1 4 5 7 12	6
2 1 5 1	1

Задача 2. Эффективная быстрая сортировка

Ограничение по времени: 3 с. Ограничение по памяти: 64 Mb.

Тимофей решил организовать соревнование по спортивному программированию, чтобы найти талантливых стажёров. Задачи подобраны, участники зарегистрированы, тесты написаны. Осталось придумать, как конце соревнования будет определяться победитель.

Каждый участник имеет уникальный логин. Когда соревнование закончится, к нему будут привязаны два показателя: количество решённых задач P_i размер штрафа F_i, Штраф начисляется за неудачные попытки и время, затраченное на задачу.

Тимофей решил сортировать таблицу результатов следующим образом: при сравнении двух участников выше будет идти тот, у которого решено больше задач. При равенстве числа решенных задач первым идет участник с меньшим штрафом. Если же и штрафы совпадают, то первым будет тот. у которого логин идёт раньше в алфавитном (лексикографическом) порядке.

Тимофей заказал толстовки для победителей и накануне поехал за ними в магазин. В своё отсутствие он поручил вам реализовать алгоритм быстрой сортировки (англ. quick sort) для таблицы результатов. Так как Тимофей любит спортивное программирование и не любит зря расходовать оперативную память, то ваша реализация сортировки не может потреблять O(n) дополнительной памяти для промежуточных данных (такая модификация быстрой сортировки называется "in-place”).

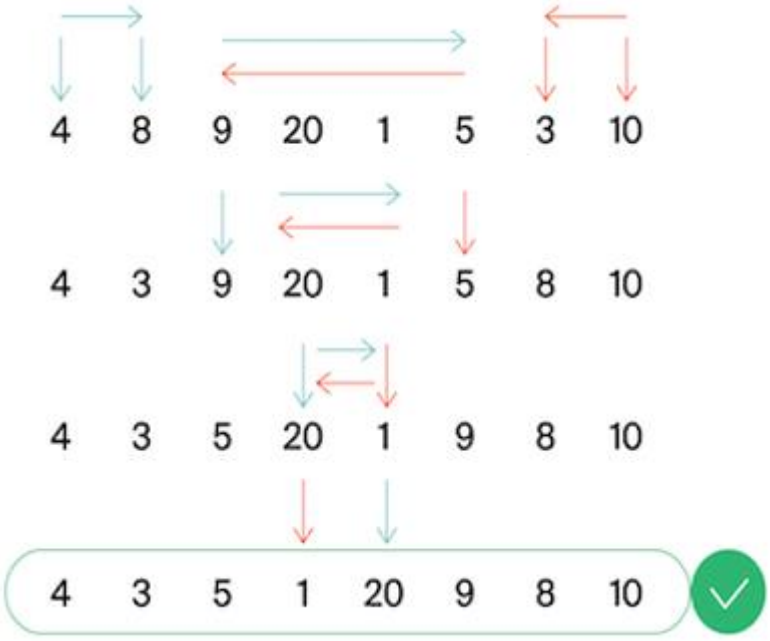
Как работает in-place quick sort

Как и в случае обычной быстрой сортировки, которая использует дополнительную память, необходимо выбрать опорный элемент (англ. pivot), а затем переупорядочить массив. Сделаем так, чтобы сначала шли элементы, не превосходящие опорного, а затем — большие опорного.

Затем сортировка вызывается рекурсивно для двух полученных частей. Именно на этапе разделения элементов на группы в обычном алгоритме используется дополнительная память. Теперь разберёмся, как реализовать этот in-place.

Пусть мы как-то выбрали опорный элемент. Заведем два указателя left и right, которые изначально будут указывать на левый и правый концы отрезка соответственно. Затем будем двигать первый указатель вправо до тех пор. пока он указывает на элемент, меньший опорного. Аналогично двигаем правый указатель влево, пока он стоит на элементе, превосходящем опорный. В итоге окажется, что что левее от left все элементы точно принадлежат первой группе, а правее от right — второй. Элементы, на которых стоят указатели, нарушают порядок. Поменяем их местами (в большинстве языков программирования используется функция swap() и продвинем указатели на следующие элементы. Будем повторять это действие до тех пор, пока left и right не столкнутся.

На рисунке представлен пример разделения при pivot=5. Указатель left — голубой, right — оранжевый.



Формат входных данных:

В первой строке задано число участников n, $1 \leq n \leq 100000$.

В каждой из следующих n строк задана информация про одного из участников.

i-й участник описывается тремя параметрами:

- уникальным логином (строкой из маленьких латинских букв длиной не более 20)
- числом решённых задач P_i
- штрафом F_i

F_i и P_i – целые числа, лежащие в диапазоне от 0 до 10^9.

Формат выходных данных:

Для отсортированного списка участников выведите по порядку их логины по одному в строке.

Примеры:

Стандартный ввод	Стандартный вывод
5 alla 4 100 gena 6 1000 gosha 2 90 rita 2 90 Timofey 4 80	gena timofey alla gosha rita
5 alla 0 0 gena 0 0 gosha 0 0 rita 0 0 Timofey 0 0	alla gena gosha rita timofey

Задача 13. Максимальная тройка

Ограничение по времени: 1.5 с. Ограничение по памяти: 8 Mb.

Имеется не более 1000000 целых чисел, каждое из которых лежит в диапазоне от -1000000 до 1000000. Найти максимально возможное значение произведений любых трех различных по номерам элементов массива.

Формат входных данных:

N

A1

A2

…

AN

Формат выходных данных:

MaxPossibleProduct

Примеры:

Стандартный ввод	Стандартный вывод
10 -1 2 3 -4 -2	75

5 -1 5 -3 -2	

Задача 14. Сортировка по многим полям

Ограничение по времени: 2 с. Ограничение по памяти: 64 Mb.

В базе данных хранится N записей, вида (Name, a_1, a_2, …, a_k) – во всех записях одинаковое число параметров. На вход задачи подается приоритет полей – перестановка на числах 1, …, k – записи нужно вывести по невозрастанию в соответствии с этим приоритетом. В случае, если приоритет полей таков: 3 4 2 1, то это следует воспринимать так: приоритет значений из 3 колонки самый высокий, приоритет значений из колонки 4 ниже, приоритет значений из колонки 2 еще ниже, а приоритет значений из колонки 1 самый низкий.

Формат входных данных:

$N \leq 1000$

$k: 1 \leq k \leq 10$

p_1 p_2 … p_k – перестановка на k числах, разделитель – пробел

N строк вида

Name a_1 a_2 … a_k

Формат выходных данных:

N строк с именами в порядке, согласно приоритету

Примеры:

Стандартный ввод	Стандартный вывод
3 3 2 1 3 A 1 2 3 B 3 2 1 C 3 1 2	B A C

Замечание. Так как колонка под номером 2 самая приоритетная, то переставить записи можно только двумя способами: (A, B, C) и (B, A, C). Следующий по приоритетности столбец – первый, и он позволяет выбрать из возможных перестановок только (B, A, C). Так как осталась ровно одна перестановка, третий приоритет не имеет значения.

Задача 15. Оболочка.

Ограничение по времени: 2 с. Ограничение по памяти: 64 Mb.

Имеется массив из N целочисленных точек на плоскости.

Требуется найти периметр наименьшего охватывающего многоугольника, содержащего все точки.

Формат входных данных:

N

x1 y1

x2 y2

…

xn yn

$5 \le N \le 500000$

$-10000 \le x_i, y_i \le 10000$

Формат выходных данных:

Одно вещественное число – периметр требуемого многоугольника с двумя знаками после запятой.

Примеры:

Стандартный ввод	Стандартный вывод
5 2 1 2 2 2 3 3 2 1 2	5.66

Задача 16. Очень быстрая сортировка.

Ограничение по времени: 1.5 с. Ограничение по памяти: 512 Mb.

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

$A1 = K$

$A_{i+1} = (A_i \times M) \% (2^{32} - 1) \% L$

Требуется найти сумму всех нечетных по порядку элементов в отсортированной по неубыванию последовательности по модулю 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 {10; 91; 83; 79; 27}.

Отсортированная последовательность {7; 27; 79; 83; 91}.

Сумма элементов на нечетных местах = (7 + 79 + 91)%100 = 77.

Формат входных данных:

N K M L

$5000000 \le N \le 60000000$, $0 \le K, L, M \le 2^{32} - 1$

Формат выходных данных:

RESULT

Примеры:

Стандартный ввод	Стандартный вывод
5 7 13 100	77

Замечание. Для представления элементов последовательности необходимо использовать тип данных unsigned int.

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)&0xFFFFFFFU)%L;

Внимание! Изменение типа данных и/или метода генерации элементов массива может привести (и на различных компиляторах приводит) к другой последовательности!

Задача 17. Внешняя сортировка.

Ограничение по времени: 2 с. Ограничение по памяти: 2 Mb.

В файле «input.txt» содержатся строки символов, длина каждой строки не превышает 10000 байт. Файл нужно отсортировать в лексикографическом порядке и вывести результат в файл «output.txt». Вот беда, файл занимает много мегабайт, а в Вашем распоряжении оказывается вычислительная система с очень маленькой оперативной памятью. Но файл должен быть отсортирован!

Примеры:

input.txt	output.txt
qwertyuiopasdffghhj qpoiuytredgfhfd asdfghjjklvcvx alkjghcdysdfgsr pquytrgsdjdsa akjhfgdghshhfuushvdfs	akjhfgdghshhfuushvdfs alkjghcdysdfgsr asdfghjjklvcvx pquytrgsdjdsa qpoiuytredgfhfd qwertyuiopasdffghhj

Задача 18. Музей.

Ограничение по времени: 1 с. Ограничение по памяти: 16 Mb.

В музее регистрируется в течение суток время прихода и ухода каждого посетителя. Таким образом, за день получены N пар значений, где первое значение в паре показывает время прихода посетителя и второе значение - время его ухода. Требуется найти максимальное число посетителей, которые находились в музее одновременно.

Формат входных данных:

В первой строке входного файла INPUT.TXT записано натуральное число N ($N < 10^5$) – количество зафиксированных посетителей в музее в течении суток. Далее, идут N строк с информацией о времени визитов посетителей: в каждой строке располагается отрезок времени посещения в формате «ЧЧ:ММ ЧЧ:ММ» ($00:00 \le$ ЧЧ:ММ $\le 23:59$).

Формат выходных данных:

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число — максимальное количество посетителей, одновременно находящихся в музее.

Примеры:

input.txt	output.txt
6 09:00 10:07 10:20 11:35 12:00 17:00 11:00 11:30 11:20 12:30 11:30 18:15	4

Источник задачи: здесь.

Задача 19. Охрана.

Ограничение по времени: 1 с. Ограничение по памяти: 16 Mb.

На секретной военной базе работает N охранников. Сутки поделены на 10000 равных промежутков времени, и известно когда каждый из охранников приходит на дежурство и уходит с него. Например, если охранник приходит в 5, а уходит в 8, то значит, что он был в 6, 7 и 8-ой промежуток. В связи с уменьшением финансирования часть охранников решено было сократить. Укажите: верно ли то, что для данного набора охранников, объект охраняется в любой момент времени хотя бы одним охранником и удаление любого из них приводит к появлению промежутка времени, когда объект не охраняется.

Формат входных данных:

В первой строке входного файла INPUT.TXT записано натуральное число K ($1 \le K \le 30$) – количество тестов в файле. Каждый тест начинается с числа N ($1 \le N \le 10000$), за которым следует N пар неотрицательных целых чисел A и B - время прихода на дежурство и ухода ($0 \le A < B \le 10000$) соответствующего охранника. Все числа во входном файле разделены пробелами и/или переводами строки.

Формат выходных данных:

В выходной файл OUTPUT.TXT выведите K строк, где в M-ой строке находится слово Accepted, если M-ый набор охранников удовлетворяет описанным выше условиям. В противном случае выведите Wrong Answer.

Примеры:

input.txt	output.txt
2 3 0 3000 2500 7000 2700 10000 2 0 3000 2700 10000	Wrong Answer Accepted

Источник задачи: здесь.

Задача 20. Аттракцион.

Ограничение по времени: 1 с. Ограничение по памяти: 16 Mb.

На протяжении многих лет в Байтландии существует парк развлечений “Funny byte”, в котором представлено много различных аттракционов: колесо вычислений, веселые горки с трассами в форме интегралов, равнобедренная ромашка и многие другие.

В последние годы популярность “Funny byte” стала неуклонно падать. В целях привлечения посетителей руководство парка решило открыть новый аттракцион, который представляет собой сложный механизм.

Кресла аттракциона расположены в N рядов по M кресел в каждом. То есть каждое кресло характеризуется номером ряда и номером колонки, в котором оно находится. Ряды нумеруются последовательно сверху вниз начиная с единицы, колонки нумеруются слева направо начиная с единицы. Каждое кресло имеет свой уникальный номер. Кресло, находящееся в i-м ряду и в j-ой колонке, имеет номер (i-1)*M + j.

После посадки отдыхающих, кресла поднимают вверх над землей. И начинается веселье! Механизм аттракциона случайным образом производит некоторое количество операций. Под одной операцией понимается взаимная перестановка двух рядов либо двух колонок. При взаимной перестановке двух рядов или колонок каждое кресло в ряду или колонке заменятся на соответствующее ему кресло в другом ряду или колонке.

И вот аттракцион завершил свою работу. Но есть одна трудность! Кресла надо вернуть в начальное положение при помощи таких же операций. Как количество, так и сами

операции не обязательно должны быть идентичны тем, которые производил аттракцион во время сеанса. Руководство парка решило, что вернуть кресла в начальное положение необходимо не более чем за 1000 операций.

Такая задача оказалась не по силам разработчикам механизма. Помогите им! От вас требуется разработать программу, позволяющую вернуть кресла в начальное положение. Гарантируется, что решение существует.

Формат входных данных:

В первой строке входного файла INPUT.TXT заданы два натуральных числа N и M ($1 \le N, M \le 250$). В последующих N строках задано по M натуральных чисел, где j-е число в i+1-й строке соответствует номеру кресла после окончания сеанса аттракциона. Числа в строках разделяются одиночными пробелами.

Формат выходных данных:

В первой строке выходного файла OUTPUT.TXT должно быть выведено количество операций перестановки K, которое не должно превышать 1000. Каждая из следующих K строк описывает одну операцию. Каждая операция описывается строкой вида Q X Y, где Q – символ 'R'(ASCII 82) либо символ 'C'(ASCII 67). Если Q равно 'R', то данная операция является перестановкой рядов, если Q равно 'C', то операция является перестановкой колонок. X и Y – два натуральных числа, соответствующие номерам рядов (колонок), которые будут переставлены в результате данной операции. Операции должны быть выведены в порядке осуществления, то есть последовательное применение которых позволит вернуть кресла в начальное положение.

Примеры:

input.txt	output.txt
2 2 4 3 2 1	2 C 1 2 R 1 2
3 5 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	0
4 5 10 7 9 8 6 15 12 14 13 11 20 17 19 18 16 5 2 4 3 1	5 R 1 4 C 1 5 C 3 4 R 2 4 R 3 4

Источник задачи: здесь.

Задачи по теме 3. Элементарные структуры данных

Задача 21. Дек

Ограничение по времени – 1 с. Ограничение по памяти – 64 Mb.

Гоша реализовал структуру данных Дек, максимальный размер которого определяется заданным числом. Методы push_back(x), push_front(x), pop_back(), pop_front() работали корректно. Но, если в деке было много элементов, программа работала очень долго. Дело в том, что не все операции выполнялись за O(1). Помогите Гоше! Напишите эффективную реализацию. **Внимание: при реализации нельзя использовать связный список.**

Формат входных данных

В первой строке записано количество команд n — целое число, не превосходящее 5000. Во второй строке записано число m — максимальный размер дека. Он не превосходит 1000. В следующих n строках записана одна из команд:

- push_back(value) – добавить элемент в конец дека. Если в деке уже находится максимальное число элементов, вывести «error».
- push_front(value) – добавить элемент в начало дека. Если в деке уже находится максимальное число элементов, вывести «error».
- pop_front() – вывести первый элемент дека и удалить его. Если дек был пуст, то вывести «error».
- pop_back() – вывести последний элемент дека и удалить его. Если дек был пуст, то вывести «error».

Value — целое число, по модулю не превосходящее 1000.

Формат выходныых данных

Выведите результат выполнения каждой команды на отдельной строке. Для успешных запросов push_back(x) и push_front(x) ничего выводить не надо.

Примеры:

Стандартный ввод	Стандартный вывод
4 4 push_front 861 push_front -819 pop_back pop_back	861 -819
7 10 push_front -855 push_front 720 pop_back pop_back push_back 844 pop_back push_back 823	-855 720 844

Задача 22. Миллиардеры

Ограничение времени: 3 с. Ограничение памяти: 64 Mb.

Возможно, вы знаете, что из всех городов мира больше всего миллиардеров живёт в Москве. Но, поскольку работа миллиардера подразумевает частые перемещения по всему свету, в определённые дни какой-то другой город может занимать первую строчку в таком рейтинге. Ваши приятели из ФСБ, ФБР, MI5 и Шин Бет скинули вам списки перемещений

всех миллиардеров за последнее время. Ваш работодатель просит посчитать, сколько дней в течение этого периода каждый из городов мира был первым по общей сумме денег миллиардеров, находящихся в нём.

Формат входных данных:

В первой строке записано число n — количество миллиардеров ($1 \le n \le 10000$). Каждая из следующих n строк содержит данные на определённого человека: его имя, название города, где он находился в первый день данного периода, и размер состояния. В следующей строке записаны два числа: m — количество дней, о которых есть данные ($1 \le m \le 50000$), k — количество зарегистрированных перемещений миллиардеров ($0 \le k \le 50000$). Следующие k строк содержат список перемещений в формате: номер дня (от 1 до $m - 1$), имя человека, название города назначения. Вы можете считать, что миллиардеры путешествуют не чаще одного раза в день, и что они отбывают поздно вечером и прибывают в город назначения рано утром следующего дня. Список упорядочен по возрастанию номера дня. Все имена и названия городов состоят не более чем из 20 латинских букв, регистр букв имеет значение. Состояния миллиардеров лежат в пределах от 1 до 100 миллиардов.

Формат выходных данных:

В каждой строке должно содержаться название города и, через пробел, количество дней, в течение которых этот город лидировал по общему состоянию миллиардеров, находящихся в нём. Если таких дней не было, пропустите этот город. Города должны быть отсортированы по алфавиту (используйте обычный порядок символов: ABC...Zabc...z).

Задача 23. Пирамидальная сортировка

Ограничение по времени – 1.5 с. Ограничение по памяти – 256 Mb.

В данной задаче необходимо реализовать сортировку кучей. При этом кучу необходимо реализовать самостоятельно, использовать имеющиеся в языке реализации нельзя. Сначала рекомендуется решить задачи про просеивание вниз и вверх.

Тимофей решил организовать соревнование по спортивному программированию, чтобы найти талантливых стажёров. Задачи подобраны, участники зарегистрированы, тесты написаны. Осталось придумать, как в конце соревнования будет определяться победитель.

Каждый участник имеет уникальный логин. Когда соревнование закончится, к нему будут привязаны два показателя: количество решённых задач P_i и размер штрафа F_i. Штраф начисляется за неудачные попытки и время, затраченное на задачу.

Тимофей решил сортировать таблицу результатов следующим образом: при сравнении двух участников выше будет идти тот, у которого решено больше задач. При равенстве числа решённых задач первым идёт участник с меньшим штрафом. Если же и штрафы совпадают, то первым будет тот, у которого логин идёт раньше в алфавитном (лексикографическом) порядке.

Тимофей заказал толстовки для победителей и накануне поехал за ними в магазин. В своё отсутствие он поручил вам реализовать алгоритм сортировки кучей (англ. Heapsort) для таблицы результатов.

Формат входных данных:

В первой строке задано число участников n, $1 \le n \le 100000$.

В каждой из следующих и строк задана информация про одного из участников.

i -й участник описывается тремя параметрами:

– уникальным логином (строкой из маленьких латинских букв длиной не более 20)

– числом решенных задач P_i
– Штрафом F_i

F_i и P_i - целые числа, лежащие в диапазоне от 0 до 10^9.

Формат выходных данных:

Для отсортированного списка участников выведите по порядку их логины по одному в строке.

Примеры:

Стандартный ввод	Стандартный вывод
5 alla 4 100 gena 6 1000 gosha 2 90 rita 2 90 timofey 4 80	gena timofey alla gosha rita
5 alla 0 0 gena 0 0 gosha 0 0 rita 0 0 timofey 0 0	alla gena gosha rita timofey

Задача 24. Очередь с минимумом

Реализуйте работу очереди. В дополнение к стандартным операциям очереди, необходимо также отвечать на запрос о минимальном элементе из тех, которые сейчас находится в очереди. Для каждой операции запроса минимального элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо «+ N», либо «−», либо «?». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего 109. Команда «−» означает изъятие элемента из очереди. Команда «?» означает запрос на поиск минимального элемента в очереди.

Формат входных данных:

В первой строке содержится M (1≤M≤106) — число команд. В последующих строках содержатся команды, по одной в каждой строке.

Формат выходных данных:

Для каждой операции поиска минимума в очереди выведите её результат. Результаты должны быть выведены в том порядке, в котором эти операции встречаются во входном файле. Гарантируется, что операций извлечения или поиска минимума для пустой очереди не производится.

Задача 25. Куча ли?

Структуру данных «куча», или, более конкретно, «неубывающая пирамида», можно реализовать на основе массива. Для этого должно выполнятся основное свойство неубывающей пирамиды, которое заключается в том, что для каждого 1≤i≤n выполняются условия:

- если 2i≤n, то a[i]≤a[2i];
- если 2i+1≤n, то a[i]≤a[2i+1].

Дан массив целых чисел. Определите, является ли он неубывающей пирамидой.

Формат входных данных:

Первая строка входного файла содержит целое число n ($1 \le n \le 106$). Вторая строка содержит n целых чисел, по модулю не превосходящих $2 \cdot 109$.

Формат выходных данных:

Выведите «YES», если массив является неубывающей пирамидой, и «NO» в противном случае.

Задача 26. Очередь с приоритетами

Реализуйте очередь с приоритетами. Ваша очередь должна поддерживать следующие операции: добавить элемент, извлечь минимальный элемент, уменьшить элемент, добавленный во время одной из операций.

Формат входных данных:

В первой строке входного файла содержится число n ($1 \le n \le 106$) - число операций с очередью.

Следующие n строк содержат описание операций с очередью, по одному описанию в строке. Операции могут быть следующими:

A x — требуется добавить элемент x в очередь.

X — требуется удалить из очереди минимальный элемент и вывести его в выходной файл. Если очередь пуста, в выходной файл требуется вывести звездочку «*».

D x y — требуется заменить значение элемента, добавленного в очередь операцией A в строке входного файла номер x+1, на y. Гарантируется, что в строке x+1 действительно находится операция A, что этот элемент не был ранее удален операцией X, и что y меньше, чем предыдущее значение этого элемента.

В очередь помещаются и извлекаются только целые числа, не превышающие по модулю 109.

Формат выходных данных:

Выведите последовательно результат выполнения всех операций X, по одному в каждой строке выходного файла. Если перед очередной операцией X очередь пуста, выведите вместо числа звездочку «*».

Задача 27. Постфиксная запись

В постфиксной записи (или обратной польской записи) операция записывается после двух операндов. Например, сумма двух чисел `A` и `B` записывается как `A B +`. Запись `B C + D *` обозначает привычное нам `(B + C) * D`, а запись `A B C + D * +` означает `A + (B + C) * D`. Достоинство постфиксной записи в том, что она не требует скобок и дополнительных соглашений о приоритете операторов для своего чтения.

Дано выражение в обратной польской записи. Определите его значение.

Формат входных данных:

В первой строке входного файла дано число N ($1 \le N \le 10^6$) - число элементов выражения. Во второй строке содержится выражение в постфиксной записи, состоящее из N элементов. В выражении могут содержаться неотрицательные однозначные числа и операции +, −, *. Каждые два соседних элемента выражения разделены ровно одним пробелом.

Формат выходных данных:

Необходимо вывести значение записанного выражения. Гарантируется, что результат выражения, а также результаты всех промежуточных вычислений, по модулю будут меньше, чем 231.

Задача 28. Считаем комментарии.

Ограничение по времени: 1 с. Ограничение по памяти: 256 Mb.

Комментарием в языке Object Pascal является любой текст, находящийся между последовательностью символов, начинающих комментарий определенного вида и последовательностью символов, заканчивающей комментарий этого вида.

Виды комментариев могут быть следующие:

1. Начинающиеся с набора символов (* и заканчивающиеся набором символов *).
2. Начинающиеся с символа { и заканчивающиеся символом }.
3. Начинающиеся с набора символов // и заканчивающиеся символом новой строки.

Еще в языке Object Pascal имеются литеральные строки, начинающиеся с символа одиночной кавычки ‘ и заканчивающиеся этим же символом. В корректной программе строки не могут содержать символа перехода на новую строку.

Будьте внимательны, в задаче используются только символы с кодами до 128, то есть, кодировка ASCII. При тестировании своего решения будьте внимательны. Код одиночной кавычки – 39, двойной – 34.

Формат входных данных:

На вход программы подается набор строк, содержащих фрагмент корректной программы на языке Object Pascal.

Формат выходных данных:

Выходом программы должно быть 4 числа – количество комментариев первого, второго и третьего типов, а также количество литеральных строк.

Примеры:

Стандартный ввод	Стандартный вывод

program test;
(*just for testing *)
var
(* variables
note that
// here is not comment
and (* here is
not a begin of
another comment
*)
x: integer; (* *)
begin
write(‘(*is not comment//’);
write(‘ and (*here*) ‘
,x // y);
End. // It is comment
	3 0 2 2

Задача 29. Расстояние в дереве

Дано взвешенное дерево. Найти кратчайшее расстояние между заданными вершинами.

Формат входных данных:

Первая строка содержит целое число n — количество вершин в дереве ($1 \le n \le 50000$). Вершины нумеруются целыми числами от 0 до n – 1. В следующих n – 1 строках содержится по три целых числа u, v, w, которые соответствуют ребру весом w ($0 \le w \le 1000$), соединяющему вершины u и v. В следующей строке содержится целое число m — количество запросов ($1 \le m \le 75000$). В следующих m строках содержится по два числа — номера вершин, расстояние между которыми необходимо вычислить.

Формат выходных данных:

Для каждого запроса выведите на отдельной строке одно число — искомое расстояние.

Задача 30. Вложенные отрезки

Ограничение времени: 1 секунда. Ограничение памяти: 64 МБ.

На прямой лежат n отрезков. Для каждой пары отрезков известно, что они либо не имеют общих точек, либо все точки одного из них также принадлежат и другому отрезку.

Дано m запросов. Каждый запрос представляет собой точку на прямой. Найдите для каждого запроса отрезок минимальной длины, которому принадлежит эта точка.

Формат входных данных:

В первой строке записано целое число n — количество отрезков ($1 \le n \le 10^5$). i-я из следующих n строк содержит целые числа a_i и b_i — координаты концов i-го отрезка ($1 \le a_i < b_i \le 10^9$). Отрезки упорядочены по неубыванию a_i, а при $a_i = a_j$ — по убыванию длины. Совпадающих отрезков нет. В следующей строке записано целое число m — количество запросов ($1 \le m \le 10^5$). В j-й из следующих m строк записано целое число c_j — координата точки ($1 \le c_j \le 10^9$). Запросы упорядочены по неубыванию c_j.

Формат выходных данных:

Для каждого запроса выведите номер искомого отрезка в отдельной строке. Если точка не принадлежит ни одному отрезку, выведите «-1». Отрезки пронумерованы числами от 1 до n в том порядке, в котором они перечислены во входных данных.

Задача 31. Четность

Ограничение времени: 2 секунды. Ограничение памяти: 64 МБ

Вы играете со своим другом в следующую игру. Ваш друг записывает последовательность, состоящую из нулей и единиц. Вы выбираете непрерывную подпоследовательность (например, подпоследовательность от третьей до пятой цифры включительно) и спрашиваете его, чётное или нечётное количество единиц содержит эта подпоследовательность. Ваш друг отвечает, после чего вы можете спросить про другую подпоследовательность, и так далее.

Ваша задача — угадать всю последовательность чисел. Но вы подозреваете, что некоторые из ответов вашего друга могут быть неверными, и хотите уличить его в обмане. Вы решили написать программу, которая получит наборы ваших вопросов вместе с ответами друга и найдет первый ответ, который гарантированно неверен. Это должен быть

такой ответ, что существует последовательность, удовлетворяющая ответам на предыдущие вопросы, но никакая последовательность не удовлетворяет этому ответу.

Формат входных данных:

Ввод содержит несколько тестов. Первая строка каждого теста содержит одно число, равное длине последовательности нулей и единиц. Эта длина не превосходит 10^9. Во второй строке находится одно неотрицательное целое число — количество заданных вопросов и ответов на них. Количество вопросов и ответов не превышает 5 000. Остальные строки содержат вопросы и ответы. Каждая строка содержит один вопрос и ответ на этот вопрос: два целых числа (позиции первой и последней цифр выбранной подпоследовательности) и одно слово — “`even`” или “`odd`” — ответ, сообщающий чётность количества единиц в выбранной подпоследовательности, где “`even`” означает чётное количество единиц, а “`odd`” означает нечётное количество. Ввод заканчивается строкой, содержащей −1.

Формат выходных данных:

Каждая строка вывода должна содержать одно целое число X. Число X показывает, что существует последовательность нулей и единиц, удовлетворяющая первым X условиям чётности, но не существует последовательности, удовлетворяющей $X + 1$ условию. Если существует последовательность нулей и единиц, удовлетворяющая всем заданным условиям, то число X должно быть равно количеству всех заданных вопросов.

Задача 32. Дерево

Ограничение времени: 1 секунда. Ограничение памяти: 64 МБ.

Рассмотрим дерево, состоящее из n вершин. Назовём *расстоянием* между двумя вершинами минимальное количество рёбер в пути, соединяющем эти вершины. По вершине v_i и расстоянию d_i найдите такую вершину u_i, что расстояние между v_i и u_i равняется d_i.

Формат входных данных:

В первой строке записано количество вершин n ($1 \le n \le 20000$) и количество запросов q ($1 \le q \le 50000$). Каждая из следующих $n - 1$ строк описывает ребро и содержит номера вершин, соединённых этим ребром. Вершины занумерованы числами от 1 до n. В следующих q строках заданы запросы. Каждый запрос представляет собой строку, в которой записаны числа v_i ($1 \le v_i \le n$) и d_i ($0 \le d_i \le n$).

Формат выходных данных:

Выведите q строк. В i-й строке выведите номер вершины u_i — ответ на i-й запрос. Если существует несколько возможных ответов, выведите любой из них. Если искомой вершины не существует, выведите 0.