Лабораторная работа №3

В рамках данной лабораторной 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 структуры данных и методы сортировки, использование встроенных не допускается (кроме массивов).

Отчет о выполнении практической работы должен содержать:

1. Титульный лист
2. Содержание
3. 4 параграфа, 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 лабораторная работа защищается на занятии преподавателю.

Задачи по теме 3. Элементарные структуры данных

Задача 1. 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 2. 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 3. 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, $l \le n \le 100000$.

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 4. 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 5. 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 6. 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 7. 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 8. 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

Задача9. 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 10. 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 11. 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 12. 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.