Лабораторная работа №4

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.

Отчет о выполнении практической работы должен содержать:

1. Титульный лист

2. Содержание

3. 3 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*. 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).

Задачи по теме 4. Деревья

Задача 1. Запросы сумм

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 мегабайта

В первой строке файла содержатся два числа: количество элементов в массиве V: $10 \le N \le 500000$ и количество запросов $1 \le M \le 500000$. Каждый элемент массива лежит в интервале $[0 \ldots 2^{32})$.

Каждый запрос – отдельная строка, состоящая из кода запроса, который может быть равен 1 или 2 и аргументов запроса.

Запрос с кодом один содержит два аргумента, начало L и конец отрезка R массива. В ответ на этот запрос программа должна вывести значения суммы элементов массива от V[L] до V[R] включительно.

Запрос с кодом два содержит тоже два аргумента, первый из которых есть номер элемента массива V, а второй – его новое значение.

Количество выведенных строк должно совпадать с количеством запросов первого типа.

Пример

Стандартный ввод	Стандартный вывод
10 8 1 7 15 8 9 15 15 19 5 19 1 1 8 1 6 8 1 0 6 2 6 6 2 1 6 2 0 9 1 4 7 1 3 6	93 39 70 49 38

Задача 2. Поиск множеств

Ограничение по времени: 1 секунда

Ограничение по памяти: 64 мегабайта

В первой строке файла содержится три числа: N – количество эталонных множеств, M – размер каждого из множеств и K – количество пробных множеств.

Каждое из множеств содержит целые числа от 0 до 10^9, числа могут повторяться.

Требуется для каждого из пробных множеств вывести в отдельной строке цифру ‘1’, если это множество в точности совпадает с каким-либо из эталонных множеств и цифру ‘0’, если оно ни с одним не совпадает, то есть выведено должно быть в точности K строк.

$5 \le N \le 50000$

$3 \le M \le 1000$

$5 \le K \le 50000$

Примеры:

Стандартный ввод	Стандартный вывод
10 3 5 6 5 1 7 9 3 2 3 2 7 2 9 9 6 2 6 6 6 9 4 1 8 4 4 8 3 2 1 2 6 9 7 2 1 6 5 3 7 7 4 4 6 3 9 7	1 1 0 0 1
10 7 5 8 4 0 3 6 9 2 3 5 0 4 3 1 1 7 1 0 3 1 2 4 7 1 5 1 5 5 1 3 4 0 0 3 4 0 3 3 3 6 3 9 3 3 4 1 3 1 8 1 1 1 6 8 6 8 2 5 6 8 1 3 9 3 7 5 7 1 4 0 3 1 1 3 3 8 4 1 2 1 1 6 6 8 8 1 1 5 7 5 1 5 3 4 1 3 1 1 8 0 0 1 2 8 2 6	1 1 1 1 0

Задача 3. Телефонная книга

Ограничение по времени: 2 секунды

Ограничение по памяти: 64 мегабайта

Необходимо разработать программу, которая является промежуточным звеном в реализации телефонной книги. На вход подается $N \leq 1000$ команд вида

ADD User Number

DELETE User

EDITPHONE User Number

PRINT User

Согласно этим командам нужно соответственно добавить пользователя в телефонную книгу, удалить пользователя, изменить его номер и вывести на экран его данные. В случае невозможности выполнить действие, необходимо вывести **ERROR.** Добавлять пользователя, уже существующего в телефонной книге, нельзя.

Необходимо вывести протокол работы телефонной книги

Формат входных данных:

См. пример.

Формат выходных данных:

В случае невозможности выполнения действия требуется вывести **ERROR**

В случае команды **PRINT** User

User Number

Примеры:

Стандартный ввод	Стандартный вывод
9 ADD IVAN 1178927 PRINT PETER ADD EGOR 123412 PRINT IVAN EDITPHONE IVAN 112358 PRINT IVAN PRINT EGOR DELETE EGOR EDITHPHONE EGOR 123456	ERROR IVAN 1178927 IVAN 112358 EGOR 123412 ERROR

Задача 4. Анаграммы

Ограничение по времени: 0.5 секунд

Ограничение по памяти: 256 мегабайт

Как известно, анаграммами называются слова, которые могут получиться друг из друга путем перестановки букв, например LOOP, POOL, POLO. Будем называть все слова такого рода *комплектом*.

На вход программы подается число слов $1 \leq N \leq 100000$. В каждой из очередных N строк присутствует одно слово, состоящее из заглавных букв латинского алфавита. Все слова имеют одинаковую длину $3 \leq L \leq 10000$.

Требуется определить число комплектов во входном множестве.

Формат входных данных:

N

W1

W2
…
WN

Формат выходных данных:
NumberOfComplects

Примеры:

Стандартный ввод	Стандартный вывод
8 BCB ABA BCB BAA BBC CCB CBC CBC	3

Задача 5. Кеширующий сервер

Ограничение по времени: 3 секунды

Ограничение по памяти: 64 мегабайта

Дормидонт работает в компании, которая занимается обработкой больших данных. Обрабатываемые данные находятся где-то в распределенной системе. Количество различных данных в системе ограничено и каждое данное имеет свой номер. Эти данные регулярно требуются различным клиентам и, поскольку время обращения к ним достаточно велико, для ускорения обработки информации Дормидонту поручено написать часть middlware — сервер-посредник, к которому и обращаются теперь клиенты за данными. Так как система — распределенная, а сервер — нет, все требуемые данные на сервер не помещаются, но он имеет возможность запоминать результаты своих запросов к распределенной системе. Для этого на сервере выделена ограниченная память на N запросов. Важно, что клиент не имеет возможности обращаться к распределенной системе и результат запроса к распределенной системе всегда должны оказаться на сервере.

К большой радости Дормидонта оказалось, что самые крупные и значимые клиенты всегда обращаются за одними и теми же данными в одном и том же порядке, так что у него есть последовательность запросов. Дормидонт придумал такой алгоритм, что как можно большее количество запросов исполняется из кеша сервера, без обращения к распределенной системе. Придумаете ли вы что-то подобное?

Формат входных данных:

На вход программы подается размер памяти под кеширование запросов $1 \le N \le 100000$, количество запросов $1 \le M \le 100000$ и ровно M запросов с номерами $0 \le R_i \le 10^{18}$. Количество различных номеров запросов ограничено и не превосходит 100000.

Формат выходных данных:

Требуется вывести одно число: сколько раз пришлось обратиться к распределенной системе за данными, отсутствующими в кеше. В начале работы кеш пуст.

Пример

стандартный ввод	стандартный вывод
5 15 3 1 4 1 5 9 2 6 5 3 5 8 7 9 3	9

Замечание

В приведенном примере первые три запроса произойдут к данным под номерами 3, 1 и 4, так как их нет в кеше. Следующий запрос, 1, есть в кеше и обращения к распределенной системе не произойдет. Запросы 5 и 9 занесут их в кеш. Следующий запрос — 2 — в кеше отсутствует, но мы выкинем из кеша запрос 1 и запрос 2 займет его место. Далее, запрос 6 вытеснит из кеша значение 2(у нас есть информация о дальнейших запросах и из нее мы видим, что запрос под номером 2 больше не повториться и нет причин хранить его далее), после чего следующие три запроса удовлетворятся из кеша. Затем произойдет еще два вытеснения — 8 и 7. Итого 9 обращений к распределенной системе. Нетрудно установить, что меньше сделать нельзя.