```
 ------------------------------------------------------------
|    Файл с данным текстом можно скопировать с помощью команды  |
|                 cp /home/valedin/task2.txt .                   |
 ------------------------------------------------------------
```

Задачи для 2 семестра

Зачет за работу во втором семестре предполагает
выполнение заданий в рамках следующих четырех этапов:

1. реализация модельной базы данных с определенной внутренней структурой и некоторым
   примитивным языком запросов (в чем-то напоминает SQL, но гораздо проще);
2. реализация сетевого взамодействия с данной базой на основе
   технологии клиент--сервер с использованием socket-интерфейса;
3. представление результатов работы с базой в виде HTML файлов,
   представление документации к реализованным программам в виде TeX файлов.
4. теоретический зачет по материалу лекций в письменном виде.

База должна поддерживать следующие операции:

--- загрузка набора данных из текстового файла;
--- сохранение полного набора данных в текстовом файле;
--- добавление отдельной записи;
--- удаление отдельной записи;
--- изменение (редактирование) записи;
--- формирование выборки на основе заданного запроса;
--- повторная выборка из уже выбранных данных;
--- представление результатов выборки в заданном формате
    (например, в виде текстового или HTML файла).

Тестирование программы должно включать

--- загрузку достаточно большого количества записей
    (сотни тысяч), сгенерированных автоматически;
--- проверку работы путем выполнения сценария (последовательности
    запросов к базе), записанного в отдельном файле;
--- одновременный запуск сервера и нескольких клиентов, каждый
    из которых работает по своему сценарию.

Конкретные варианты баз описываются ниже. Внутренняя реализация каждого варианта
представляет собой некоторую комбинацию деревьев, хеш-множеств, списков и т.п.

Предлагается установить следующие контрольные точки:

0: конец февраля --- определение и фиксация внутреннего представления данных.
1: конец марта --- база должна работать без сетевого взаимодействия.
2: конец апреля --- реализация сетевой работы, сервер поддерживает базу и отвечает на
   запросы, клиент общается с пользователем или с файлом сценария и посылает
   запросы серверу.
3: до 15 мая --- описание программы (алгоритмы, структуры внутреннего представления,
   описание языка запросов и т.п.) в виде TeX файла, выдача результатов выборок в виде HTML файла.

При злостном нарушении сроков сдачи студенту могут быть выданы дополнительные задачи
по сетевому программированию (некоторый список таких задач у меня есть) или по темам
предыдущих семестров 1 и 2 курсов.

Далее приводятся примеры задач на построение модельной базы данных. Эти задачи не догма,
их можно модифицировать, но общую идею все же следует сохранить --- использование не
совсем тривиальных структур для хранения данных, использование языка запросов для общения
с базой и последующий перенос реализации сетевое взаимодействие клиент-сервер.

==========

База ‘‘Студент’’

Модель данных.
Основным элементом данных этой базы является информация
об одном отдельно взятом студенте. Например,

```
    char   name[64];    //  ключ:  Фамилия, Имя, Отчество
    int    group;       //  ключ:  группа
    double rating;      //  ключ:  средний балл
    char * info;        //  доп. информация, например, адрес
                        // (длина не фиксирована)
```

Поля, помеченные словом "ключ" могут использоваться для поиска и формирования выборки.
Внутренее представление базы предполагает реализацию некоторых структур (деревьев, списков,
множеств), которые обеспечивали бы эффективный поиск и выборку требуемых записей. При этом

описанные выше поля могут быть собраны в некоторую структуру, снабженную дополнительными указателями ссылок по типу дерева или списка.

Для определения выборки используется упрощенный язык запросов, например, следующего вида

```
select <критерии> end     --- произвести выборку по указанным критериям
reselect <критерии> end   --- произвести выборку из уже выбранных записей
print <список полей> [sort <поле>] end  --- вывести результат выборки
insert <список значений> end  --- добавить новую запись
remove <список значений> end  --- удалить запись
и т.п.
```

Задание критериев выборки может иметь вид конкретного значения, диапазона и (дополнительно) перечисления значений или диапазонов. Например

```
select name=Ив* group=201-203 rating=3-* end
print  name rating info sort name end
```

--- выбрать всех студентов с фамилиями на "Ив" из 201, 202 и 203 групп с рейтингом большим или равным 3, из полученной выборки напечатать имена, рейтинг и дополнительную информацию, выдачу отсортировать по именам.

Интерфейс базы должен содержать функции, получающие запрос в виде текстовой строки и возвращающие результат также в виде текстового массива. Во второй части задания эти функции составят основу сетевого взаимодействия клиента и сервера.

Варианты заданий различаются способом внутреннего представления.

Задание S1.

Записи о каждой отдельной группе хранятся в виде сбалансированного дерева по полю name и упорядоченного списка по полю rating. Указатели на корни этих деревьев и начальные элементы списков хранятся в хеш-множестве по номеру группы (поле group).

Задание S2.

Записи о каждой отдельной группе хранятся в виде красно-черного дерева по полю name и обычного дерева поиска по полю rating. Указатели на корни этих деревьев хранятся в хеш-множестве по номеру группы (поле group).

Задание S3.

Записи о каждой отдельной группе хранятся в виде B-дерева по полю name и обычного дерева поиска по полю rating. Указатели на корни этих деревьев хранятся в хеш-множестве по номеру группы (поле group).

Задание S4.

Все записи связаны по полю name в сбалансированное дерево. Кроме этого существует хеш-множество, дающее список группы по номеру группы и хеш-множество, дающее список записей с данным диапазоном рейтинга (рейтинг изменяется от 0 до 5 и разбивается на классы эквивалентности с шагом 0.1).

Задание S5.

Существует хеш-множество, дающее список группы по номеру группы. Кроме этого все записи связаны в красно-черное дерево по полю name и красно-черное дерево по полю rating.

Задание S6.

Все записи хранятся в отдельном динамическом массиве. Кроме этого существуют три сбалансированных дерева, хранящие индексы этих записей и упорядоченные соответственно по полям name, rating, group.

Задание S7.

Все записи хранятся в отдельном динамическом массиве и упорядочены в нем по полю name. Кроме этого существуют два хеш-множества, дающие для номера группы список индексов соответствующих записей и для диапазона рейтинга длиной 0.1 --- список индексов записей с соответсвующим рейтингом. Поиск по имени производится методом деления пополам.

==========

База ‘‘Расписание’’

Модель данных.
Отдельная запись по расписанию содержит следующую информацию

- время (день, пара)
- аудитория
- предмет
- преподаватель

    группа (поток для лекций)

Формат представления каждого поля можно определить по своему усмотрению.

Требуется обеспечить следующие операции с этой базой:

    --- загрузка базы из текстового файла;
    --- сохранение базы в текстовом файле;
    --- добавление нового пункта в расписание
        (с проверкой возможных накладок);
    --- удаление пункта из расписания;
    --- получение частных расписаний (выборок)
        по отдельной группе, преподавателю, предмету в рамках
        набора групп (потока), аудитории (когда свободна,
        когда и чем занята), получение списка аудиторий,
        свободных в указанное время и т.п.

Для формулировки запросов следует использовать специальный язык, в котором задаются
действия и критерии выборки. Например, запрос

```
select teacher=Иванов group=100-499 subject=Analysis end
print teacher group date_time room sort group end
```

составит расписание Иванова на 1--4 курсах по анализу и напечатает его в виде списка
с указанными полями и отсортированного по номерам групп.

Операциями запроса могут быть

```
select <критерии> end     --- произвести выборку по указанным критериям
reselect <критерии> end   --- произвести выборку из уже выбранных записей
print <список полей> [sort <поле>] end  --- вывести результат выборки
insert <список значений> end  --- добавить новую запись
remove <список значений> end  --- удалить запись
и т.п.
```

Форма задания критериев аналогична критериям для базы ‘‘Студент’’.

Интерфейс базы должен содержать функции, получающие запрос в виде текстовой строки и
возвращающие результат также в виде текстового массива. Во второй части задания эти функции
составят основу сетевого взаимодействия клиента и сервера.

Варианты заданий различаются способом внутреннего представления.

Задание R1.

Основой представления является (разреженная) матрица, строки которой соответствуют дню и
времени, а столбцы --- группе (как на 15 этаже), содержимое ячейки этой матрицы определяет
преподавателя, предмет и аудиторию. Кроме этого каждый отдельный преподаватель и предмет
связаны в свой список, доступ к которому определяется из хеш-множества (что позволяет
быстро находить расписание преподавателя или предмета).

Задание R2.

Основой представления является (разреженная) матрица, строки которой соответствуют дню и
времени, а столбцы --- аудитории (как у диспетчера), содержимое ячейки этой матрицы
определяет преподавателя, предмет и группу. Кроме этого каждый отдельный преподаватель и
предмет связаны в свой список, доступ к которому определяется из хеш-множества (что
позволяет быстро находить расписание преподавателя или предмета).

Задание R3.

В базе хранится несколько массивов: A --- массив преподавателей, B --- массив предметов,
C --- массив аудиторий, D --- массив групп, E --- массив времен, а также
Матрицы соответствий AB, AC, AD, AE, BC, BD, BE, CD, CE, DE. Они содержат 0 и 1.
Например, матрица AB(i,j)=1, если i-й преподаватель ведет занятия по j-му предмету.
Аналогично матрица CD показывает занимает ли некоторая группа когда-либо конкретную
аудиторию. Эти матрицы --- по сути битовые множества, а выборка получается умножением
этих матриц на соответствующие битовые векторы.

Задание R4.

В базе хранятся:
единый динамический массив всех записей (преподаватель, группа, время, ауд., предмет),
список индексов по преподавателям (т.е. для каждого преподавателя ---
список индексов тех записей, где он встречается),
список индексов по группам,
список индексов по аудиториям,
список индексов по временам,
список индексов по предметам.

Выборка сводится в объединению/пересечению числовых множеств индексов.

==========

База ‘‘Нагрузка’’

Модель данных.
Есть множество работников, организованных в некоторую иерархию, имеющую структуру произвольного дерева с поименованными вершинами (например, начальник, заместитель, руководитель группы, исполнитель и т.п.) Есть множество производственных обязанностей (нагрузок), каждая из которых имеет свое имя и список трудозатрат.

Требуется построить базу, позволяющую добавлять, удалять, редактировать обязанности сотрудника и выдавать сводные таблицы суммарных трудозатрат для групп сотрудников в соответствии с деревом иерархий (фактически в виде суммы по вершинам, слоям или поддеревьям дерева иерархий).

В качестве конкретного примера можно взять модель данных для определения учебной нагрузки преподавателя мехмата.

Иерархия:
    факультет
    кафедра
    заведующий, профессор, доцент, ст.преп., преп., ассистент
    конкретные люди-сотрудники

Виды нагрузки:
    Лекции, семинары, лабораторные, зачеты, экзамены, консультации,
    контр.раб., вст.экз, курс.работы, дипл.работы, аспиранты, стажеры
    и т.п.

Для каждого учебного предмета известно его распределение по видам нагрузки и величина в часах занятий за семестр или год:
    прогр_1_курс_1_сем : семинары 16ч, лабораторные 32ч, зачет 8ч.
    прогр_1_курс_2_сем : семинары 16ч, лабораторные 32ч, зачет 8ч.
    прогр_2_курс_3_сем : лабораторные 32ч, экзамен 6ч,
    прогр_2_курс_4_сем : лабораторные 32ч, зачет 8ч,
    прогр_лекции_3_сем : лекции 48ч, экзамен 30ч.
    руководство курсовыми работами 1 чел : курс.работы 15ч.
    руководство аспирантами 1 чел : аспиранты 50 ч.
    и т.д. для каждого учебного предмета.

Для каждого сотрудника определен список его обязанностей в виде списка имя -- обязанности -- количество. Например,
    Валединский_В.Д,
        прогр_1_курс_1_сем 1
        прогр_1_курс_2_сем 1
        прогр_2_курс_3_сем 1
        прогр_2_курс_4_сем 1
        прогр_лекции_3_сем 1
        прогр_лекции_4_сем 1
        руководство курсовыми работами 9
        руководство аспирантами 3
        и т.д.

Требуется формировать суммарные таблицы на основе выборок.
Формат таблицы описывается в языке запросов. Например,

```
columns <список>
row for all <узлы иерархии>
row sum for <узлы иерархии>
```

Например, выборочную таблицу нагрузки доцентов кафедры вычислительной математики вида

имя	лекции	семинары	экзамены	всего
Валединский_В.Д.	80	32	36	146
Староверов В.М.	32	96	12	140
Богачев_К.Ю.	64	32	6	104
.....				
Всего	...	...	...	...

можно получить по запросу

```
columns node_name лекции семинары экзамены row_sum
row for all доцент
row sum for доцент
```

Варианты заданий различаются способом внутреннего представления.

Задание N1.

Иерархия представляется в виде произвольного дерева, где концевые вершины
соответствуют сотрудникам, а остальные вершины --- узлам иерархии (должности,
подразделения и пр.). Учебные нагрузки хранятся в виде массива пар
<название нагрузки><строка с описанием составляющих ее часов>
Каждый сотрудник хранит список, в котором содержатся индексы его нагрузок
из данного массива, количество по каждому типу и описание дополнительной информации
(например, номера групп, фимилии студентов и т.п.). Запрос обрабатывается
путем обхода дерева, выбора требуемых узлов и подсчета суммарных характеристик.

Задание N2.

Виды часов нагрузки представлены массивом заданного фиксированного размера.
Описания учебных предметов хранятся в виде массива.
Для каждого учебного предмета хранится его название и соответствующий
ему массив часов. Преподаватели также хранятся в виде массива.
Для каждого преподавателя хранится его имя, его место в иерархии, список
его учебной нагрузки в виде индексов учебных предметов.
Место в иерархии задается битовым множеством принадлежности к той или
иной категории --- кафедре, должности и т.п. Запрос обрабатывается
просмотром списка преподавателей и сравнением места в иерархии с требуемым
в запросе.

==========

База ‘‘Библиотека’’

Модель данных.
Есть информация об отдельных литературных произведенияхх, включающая в себя
название, автора, и др. выходные данные (например, издательство, объем, тираж),
а также принадлежность данного издания к некоторым категориям в соответствии с
заданными классификаторами. Классификаторов может быть несколько. Например, по теме
или области (история, математика, политика, искусство и т.п.) или по жанру (романы,
рассказы, критика, фантастика). Классификаторы могут быть древовидными (иерархическими).
Требуется построить базу, которая позволяет добавлять, удалять, редактировать записи,
а также получать выборки по заданным критериям. Например, выбрать все исторические
романы А.С.Пушкина.
Итак, основным элементом базы данных является запись, включающая
в себя информацию об отдельном издании, например, такую

```
char author[64];
char title[128];
char publisher[64]
int  class_1;      // коды по 1,2,3 классификаторам
int  class_2;
int  class_3;
а также поля для любой другой необходимой информации
```

Форма запроса к выборке аналогична запросам к базе ‘‘Студент’’, т.е. можно выбирать по
фамилии автора, издательству, названию, кодам классификатора и др.

Будем считать, что наиболее частыми являются запросы по фамилии автора и кодам классификатора.

Задание L1.

Все записи размещены в динамическом массиве, упорядоченном по фамилии автора. Кроме
этого имеется хеш-множество, которое заданному набору кодов классификаторов сопоставляет
список индексов для записей, имеющих данное хеш-значение по кодам классификаторов.

Задание L2.

Имеется разреженная матрица в которой индекс строки соответствует хеш-значению по фамилии
автора, а индекс столбца соответствует хеш-значению по вектору кодов классификатора.
Элементом этой матрицы является указатель на начало списка записей, относящихся к данному
классу эквивалентности по этим хеш-функциям.

==========

База ‘‘Торговля’’

Модель данных.

Торговая компания покупает и продает разнообразные товары.
Для каждой сделки она хранит следующую информацию:
    название товара,
    кому продано (или у кого куплено),
    количество товара,
    цена товара,
    дата сделки.

Требуется реализовать базу данных, позволяющую добавлять, удалять, редактировать записи,
а также получать сводные таблицы с балансом затрат и доходов. Например, распечатать все
сделки с указанной фирмой за указанный период времени и подсчитать суммарный баланс или выдать
всех покупателей указанного товара с указанием суммарного объема и стоимости проданного им товара.

Форма запроса к выборке аналогична запросам к базе ‘‘Студент’’ с добавлением необходимых
ключевых слов для указания о том, какие поля суммировать при выводе.

Задание M1.

В двух отдельных динамических массивах хранятся наименования товаров
и названия партнеров (кому продаем и у кого покупаем). Эти имена в
обоих массивах упорядочены в лексикографическом порядке (бинарный поиск).
В ячейках разреженной матрицы хранится цена, дата и количество каждой сделки.
Индексы элементов матрицы соответствуют индексам в динамических массивах.

Задание M2.

В ячейках разреженной матрицы хранится цена, дата и количество каждой сделки.
Строки матрицы соответствуют наименованию товара, столбцы --- названию
фирмы-партнера. В двух отдельных хеш-множествах хранятся наименования товаров
и названия партнеров (кому продаем и у кого покупаем) вместе с индексами
соответствующих строк или столбцов матрицы сделок.

==========

База ‘‘Производственый склад’’

Модель данных.
Склад хранит детали для сборки различных устройств (скажем, компьютеров).
Есть наименования деталей, и сборочные карты, которые описывают какие
детали и в каком количестве нужны для сборки определенного устройства.
Собранное устройство опять хранится на том же складе. Собранное устройство
может являться составной частью для сборки другого более сложного устройства.

Требуется определять по запросу возможна ли сборка заданного количества
заданных устройств, если нет, то каких деталей и в каком количестве не
хватает. Другой запрос --- сколько указанных устройств можно собрать из
имеющихся деталей и какие детали после сборки окажутся в дефиците.
Также нужно выполнять операции "сборки", добавления и удаления деталей.
Кроме этого нужно добавлять новые детали и устройства с их картами сборки и
распечатывать карты сборки в удобном для восприятия виде.

Задание W1.

Имена всех деталей и устройств вместе с их текущим количеством хранятся
в динамическом массиве.
Кроме этого, устройства хранятся в дереве, упорядоченном по названию устройства.
Каждому узлу дерева сопоставлена карта сборки в виде списка требуемых деталей
(указателей на место детали в массиве деталей).

Задание W2.

Имена всех деталей и устройств хранятся в одном динамическом массиве.
Вместе с именами хранится количество деталей и список индексов для составных
частей устройства. Кроме этого для быстрого поиска есть хеш-множество,
сопоставляющее имени устройства или детали его индекс в этом массиве.

==========

База ‘‘Футбол’’

Модель данных. Хранится информация об игроках (фамилия, стиль),
командах и матчах. Каждый матч описывается следующими данными:
дата матча, какие команды играли и их составы, кто на какой минуте забивал,
замены.

Примерные запросы. Добавление удаление данных, список игроков, забивших
мяч после указанной минуты матча (не важно какого), самые результативные
игроки,