1. Операция вставки в контейнер std::set имеет сложность O(log(N)). Сгенерируйте N случайных чисел и добавьте их в данный контейнер. Совокупность этих операций имеет сложность O(log(N)). Далее создайте последовательный контейнер, добавьте в него те же N случайных чисел и отсортируйте его с помощью алгоритма std::sort. Совокупность этих операций также имеет сложность O(Nlog(N)). Определите, что выполняется быстрее. В качестве последовательного контейнера можно использовать std::vector или std::array.

2. Исследуйте равномерность хэш-функции (файл collisions.cpp). Создайте большое количество случайных экземпляров данной структуры и для каждого вычислите хэш-код. Постройте график зависимости числа коллизий от количества экземпляров.

3. Сравните эффективность различных хэш-функций (файл compare_collisions.cpp). Описание функций можно посмотреть здесь http://www.partow.net/programming/hashfunctions/#AvailableHashFunctions. Сравнить свои результаты (какие из хэш-функций эффективнее) можно здесь http://vak.ru/doku.php/proj/hash/efficiency.

4. Написать контейнер для хранения записей телефонного справочника (в структуре два поля — имя и телефон). Разные клиенты собираются использовать справочник по-разному.

Городская типография собирается напечатать справочник и ей нужны записи в отсортированном по фамилии человека порядке. (использовать ordered_non_unique для поля с именем)

Рекламное агенство нуждается в произвольном доступе к записям справочника (использовать random_access<>)

Пользователь хочет за максимально короткое время находить нужную ему запись (использовать hashed_non_unique).

Напишите демонстрационный код с использованием всех интерфейсов данного контейнера.