Домашние задания по C++ курсу:

1. Основы C++ (лекции 1-3).

1.1 Ввести с консоли целое число n. Распечатать все простые числа до n включительно. Например, если n=19 или 22, то выводить 2, 3, 5, 7, 11, 13, 17, 19.

1.2 Ввести 8-байтовое целое число без знака (unsigned long long). Внести каждый из его 8 байтов в вектор байтов vector<uint8_t>. Затем распечатать каждый байт в шестнадцатиричном виде. Подсказка: удобно использовать операцию битового сдвига >> .

1.3 (Для умных) Написать Quine -- программу которая печатает собственный текст. Проверить результат командой diff (Linux) или fc (Windows).

1.4 Написать функции
char toUpper(char c)
char toLower(char c)
Которые переводят символ с в верхний или нижний регистр (только обычная латиница). Не использовать стандартные функции C/C++ для изменения регистра, использовать числовые ASCII коды. Проверить результат.

1.5 Написать свой пример программы которая содержит модуль (пару хэдер-cpp, например файлы fun.h, fun.cpp), которые реализуют минимум 2 функции, а также класс в виде 2-х отдельных файлов (Например MyClass.h, MyClass.cpp). Поместить все это в namespace Fun. Также написать файл main.cpp, с функцией main(), которая вызывает функции из fun и использует класс MyClass. Собрать это все с помощью CMake.

2. Классы. Наследование. (Лекция 4)

2.1 Написать свой пример класса в 2 файлах : MyClass.h, MyClass.cpp (например). Класс должен содержать не менее 2х приватных полей, геттеры и сеттеры на все поля, не менее 2х конструкторов, еще минимум 2 обычных метода, статическое поле и статический метод, деструктор, а также friend функцию. Реализовать часть методов в .h, а часть -- в .cpp. Продемонстрировать использование этого класса в функции main().

2.2 Написать пример архитектуры с наследованием: интерфейс или абстрактный класс, минимум 2 его потомка, минимум 2 потомка потомка. Определить (как абстрактные в предке), реализовать в потомках и переопределить не менее 2х виртуальных методов. Классы (кроме, возможно, общего предка) должны содержать protected поля, которые инициализирует конструктор. В качестве абстрактного класса-предка взять Monster,

Weapon, Deity или Hero (Или, для зануд, что-то вроде Person, Student, Employee). Продемонстрировать в функции main().

2.3 Скопировать пример 4.2 и изменить его так, чтобы класс Person наследовал Entity и использовал его поле name. Разрешить diamond problem как в примере 4.3 и сделать так, чтобы все работало. Продемонстрировать в функции main().

3. Streams. Smart Pointers. (Лекции 5-6)

3.1 (Специально для одного знакомого любителя CSV, JSON) Написать программу которая парсит входной файл в формате CSV (какой-то конкретный, не произвольный CSV), а на выходе выдает JSON (массив объектов, 1 строка = объект). Корректно обрабатывать лишние пробелы, или полное отсутствие пробелов. Пользоваться только istream, ofstream, istringstream, ostringstream, string, никаких сторонних библиотек !

3.2 Противоположная операция -- парсить JSON полученный в примере 3.1 (массив объектов) и выдавать CSV. Игнорировать пробелы и символы новой строки.

3.3 Написать пример чтения файла конфигурации в виде файла (properties), каждая строка имеет вид:
key = value
Игнорировать пустые строки, пробелы, комментарии (начинаются с # или !).
Допустить возможность заключения строковых значений в кавычки.

3.4 Написать свой пример жизненного цикла unique_ptr (например unique_ptr<string>) : функция source, функция обработки значения, и функция sink (см пример 6.1)

3.5 Реализовать класс - двунаправленный список (doubly linked list) с использованием shared_ptr и weak_ptr. Реализовать методы find, insert, erase, size. Позицию можно указывать как shared_ptr<Node>. Продемонстрировать работу. Содержимое может быть типа string или int например. В классе узла (Node) создать деструктор, который печатает значение удаляемого узла (как в Tjej), таким образом проверить что все узлы действительно умирают при смерти списка (И нет зацикленных shared_ptr !). Для умных: реализовать класс-итератор для списка, проверить работу с range for и for с итератором. Создать итераторные версии find, insert, erase.