Список задач на классы

Внимание. Для каждого класса должно быть перегружено несколько операций (либо как методы класса либо при помощи отдельных функций). Классы должны быть объявлены в заголовочных файлах. Реализацию "длинных" методов (больше одной команды в теле метода) нужно писать в файлах исходного кода. Обработку ошибок нужно реализовать при помощи исключений.

1. Реализовать класс `R3Vector`, представляющий вектор в трёхмерном пространстве. В числе прочих должны быть реализованы:

   - Конструкторы;
   - Арифметические операции `+`, `+=`, `-`, `-=`, скалярное произведение `*` и умножение на число.
   - Векторное произведение, например, `&`.
   - Операции сравнения `==`, `!=`, `<`, `<=`, `>`, `>=`.
   - Операция `>>` для класса `std::istream` и операция `<<` для класса `std::ostream` для ввода/вывода строки.
   - Должен быть метод для вычисления длины вектора.

   К обычным операциям добавляется векторное произведение. В качестве теста написать с использованием этого класса консольную программу, которая

   - находит расстояние от точки до плоскости (плоскость задается тремя точками);
   - находит угол между двумя векторами в трехмерном пространстве;
   - находит расстояние между двумя скрещивающимися прямым (прямая задается точкой и направляющим вектором).

2. Реализовать класс `Real` (вещественное число), использующий представление вещественного числа с фиксированной десятичной точкой. Число представляется с точностью до 10^{-3} в виде $M \cdot 10^{-3}$, где M — целое число. Реализация всех методов должна использовать исключительно целочисленную арифметику. В числе прочих должны быть реализованы

   - Конструкторы;
   - Преобразование числа в строку и строки в число.
   - Арифметические операции `+`, `+=`, `-`, `-=`, `*`, `*=`, `/`, `/=`.
   - Операции сравнения `==`, `!=`, `<`, `<=`, `>`, `>=`.
   - Вычисление квадратного корня.
   - Операция `>>` для класса `std::istream` и операция `<<` для класса `std::ostream` для ввода/вывода строки.

   В качестве тестовой программы написать программу решения квадратного уравнения, корни находятся с точностью 10^{-3}.

3. Реализовать класс `Polynomial`, представляющий многочлен произвольной степени с операциями сложения, умножения и получения степени, а также деления с остатком и вычисления НОД двух многочленов. В числе прочих должны быть реализованы

   - Правильные конструкторы (в том числе и конструктор копирования), деструктор и операция присваивания `=`.
   - Арифметические операции `+`, `+=`, `-`, `-=`, `*`, `*=`.
   - Операции сравнения `==`, `!=`, `<`, `<=`, `>`, `>=`.
   - Деление с остатком.

- НОД двух многочленов.
- Возведение многочлена в степень.
- Операция `>>` для класса `std::istream` и операция `<<` для класса `std::ostream` для ввода/вывода строки.

4. Реализовать класс `Set`, представляющий из себя множество целых чисел. Класс должен поддерживать операции объединения, пересечения и симметричной разности множеств, а также операции определения принадлежности элемента и операции сравнения множеств (входит ли множество в данное). В числе прочих должны быть реализованы
   - Правильные конструкторы (в том числе и конструктор копирования), деструктор и операция присваивания `=`.
   - Операции объединения `|`, `|=`, пересечения `&`, `&=` и симметричной разности множеств `^`, `^=`.
   - Операции сравнения `==`, `!=`, `<`, `<=`, `>`, `>=`. Последние четыре операции проверяют, является ли множество подмножеством другого.
   - Операции определения принадлежности элемента множеству.
   - Операция `>>` для класса `std::istream` и операция `<<` для класса `std::ostream` для ввода/вывода строки.

5. Реализовать класс `String`, являющийся оболочкой над строкой. Должны быть реализованы
   - Правильные конструкторы (в том числе и конструктор копирования), деструктор и операция присваивания `=`.
   - Операции `+` и `+=` для конкатенации строк.
   - Операции `[]` и `()` для чтения и записи отдельных символов.
   - Операции сравнения `==`, `!=`, `<`, `<=`, `>`, `>=`.
   - Операция `>>` для класса `std::istream` и операция `<<` для класса `std::ostream` для ввода/вывода строки.

6. Реализовать класс `Substitution`, представляющий подстановку порядка N, т. е. биективное отображение конечного множества из N элементов в себя. Должны быть реализованы композиция подстановок, действие подстановки на элемент X, $0 \leqslant X \leqslant N-1$, определение четности подстановки. Должны быть реализованы
   - Правильные конструкторы (в том числе и конструктор копирования), деструктор и операция присваивания `=`.
   - Композиция подстановок `*`.
   - действие подстановки на элемент `*`.
   - Определение четности подстановки.
   - Операция `>>` для класса `std::istream` и операция `<<` для класса `std::ostream` для ввода/вывода строки.
   - Напишите программу, вычисляющую определитель матрицы при помощи класса подстановок.

7. Реализовать класс `Matrix`, представляющий прямоугольную матрицу порядка $N \times N$, где N задаётся в конструкторе. Должны быть реализованы получение элемента матрицы с заданными индексами (для чтения и для записи), сумма и произведение матриц, единичная матрица, действие матрицы на вектор размерности N, транспозиция, элементарные преобразования, приведение к ступенчатому виду, вычисление определителя, обратной матрицы и т.д. Должны быть реализованы

- Правильные конструкторы (в том числе и конструктор копирования), деструктор и операция присваивания `=`.
- Арифметические операции `+`, `+=`, `-`, `-=`, `*`, `*=`.
- Операции сравнения `==`, `!=`, `<`, `<=`, `>`, `>=`.
- Операции `[]` и `()` для чтения и записи отдельных элементов.
- Операция `>>` для класса `std::istream` и операция `<<` для класса `std::ostream` для ввода/вывода строки.
- Действие матрицы на вектор размерности N, транспозиция, элементарные преобразования, приведение к ступенчатому виду, вычисление определителя, обратной матрицы и т.д.

8. Битовое множество произвольной длины `BitSet`. Для хранения битов лучше всего использовать беззнаковые типы фиксированной длины `std::uint32_t` или `std::uint64_t` из заголовочного файла `<cstdint>`. Должны быть реализованы

   - Правильные конструкторы (в том числе и конструктор копирования), деструктор и операция присваивания `=`.
   - Битовые операции `&`, `&=`, `|`, `|=`, `~`, `^`, `<<`, `<<=`, `>>`, `>>=`.
   - Операции сравнения `==`, `!=`, `<`, `<=`, `>`, `>=` как двоичных чисел.
   - Операции `[]` и `()` для чтения и записи отдельных битов.
   - Операция `>>` для класса `std::istream` и операция `<<` для класса `std::ostream` для ввода/вывода битового множества.