Задача 1

Минимум из произвольных величин

Напишите шаблонизированную функцию get_min_value. Функция должна:

- Принимать параметрами два объекта произвольного типа (гарантируется, что для них определены операции сравнения)
- Возвращать объект с минимальным значением (гарантируется, что такой объект есть)

Для тестирования можно использовать main следующего вида:

```
int main()
{
    int a = 5;
    int b = 4;
    string s1 = "abc";
    string s2 = "zxc";
    cout << "Min int: " << get_min_value(a, b) << endl;
    cout << "Min string: " << get_min_value(s1, s2) << endl;
    return 0;
}
```

На экране после этого ожидается:

```
Min int: 4
Min string: abc
```

Задача 2

Шаблонный студент

Есть шаблонная функция с вот таким прототипом:

```
template<typename T>
bool check_them(T& x, T& y, T& z);
```

Известно, что она принимает три переменных, делает с ними некие манипуляции, возвращает ответ, подходят ли они. При этом она полагается только на то, что для переданных переменных определена операция равенства (которая ==). Эта функция уже реализована и перед проверкой будет приклеена к вашему коду снизу.

Объявлены вот такие структуры (даже не классы):

```
struct student {
    std::string name;
    std::string id_number_string;
};
```

Логически студенты равны, если у них равны id_number_string. Имена могут быть при этом любыми. Допишите нужный код, чтобы функция check_them смогла проверять студентов. Например, вот так:

```
int main()
{
    student a = {"Andy", "1234 123123"};
    student b = {"Andrew", "1234 123123"};
    student c = {"Andy", "1234123123"};
    cout << boolalpha << "Check result is: " << check_them(a, b, c) << endl;
    return 0;
}
```

Задача 3

Шаблонизированный класс

В контесте к прошлой лекции был написан класс вектора на плоскости с примерно таким прототипом:

```
class Vector2D
{
public:
    // Конструкторы
    Vector2D();
    Vector2D(int x, int y);

    // Деструктор
    ~Vector2D();

    // Получение координат
    int getX() const;
    int getY() const;

    // Задание координат
    void setX(int x);
    void setY(int y);

    // Перегруженные операторы
    bool operator== (const Vector2D& v2) const;
    bool operator!= (const Vector2D& v2) const;
    Vector2D operator+ (const Vector2D& v2) const;
    Vector2D operator- (const Vector2D& v2) const;
    Vector2D operator* (const int a) const;
};

// Оператор умножения скаляра на вектор
Vector2D operator* (int a, const Vector2D& v);

// Вывод вектора, ожидается строго в формате (1; 1)
std::ostream& operator<<(std::ostream& os, const Vector2D& v);

// Чтение вектора, читает просто две координаты без хитростей
std::istream& operator>>(std::istream &is, Vector2D &v);
```

На тот момент вектор работал с целочисленными координатами. Теперь давайте шаблонизируем вектор - он должен научиться работать с координатами, хранимыми как int и как double. (Будем считать, что обычно мы хотим иметь произвольные координаты как double, но в отдельных случаях хочется работать именно с целыми числами.)

Возьмите ровно вектор из прошлого контеста (со всеми методами и перегруженными операторами), шаблонизируйте его.

Ожидается, что класс позволит выполнить примерно такой код:

```
    Vector2D<int> v1;
    cin >> v1;
    cout << "Read vector: " << v1 << endl;
    cout << "Vector multiplied by 42: " << v1 * 42 << endl;

    Vector2D<double> v2;
    cin >> v2;
```

```
    cout << "Read vector: " << v2 << endl;
    cout << "Vector multiplied by 42: " << 42 * v2 << endl;
```

И на экране после этого будет:

```
1 1                          <-- Это ввод с клавиатуры
Read vector: (1; 1)          <-- Это вывод считанного вектора int-ов
Vector multiplied by 42: (42; 42)
0.001 0.001                  <-- Ещё ввод
Read vector: (0.001; 0.001) <-- Это вывод считанного вектора double-ов
Vector multiplied by 42: (0.042; 0.042)
```

Задача 4

Размерные величины

Дан код, написанный с использованием шаблонов. Вот такой (см. Приложение)

Код реализует классы размерных величин - длины, массы, времени, скорости (и, вообще говоря, любых размерных величин, которые собираются из метров, килограммов и секунд). Код позволяет эти величины складывать и делить. При сложении и делении код следит за физической размерностью результата. Код не даст выполнить действия, если размерность результата не сходится. Также код реализует оператор вывода на экран с учётом размерности.

Код позволяет писать, например, следующее:

```
int main()
{
    // Длина
    Length l1 = {100};
    Length l2 = {200};
    // Время
    Time t = {20};

    // Скорость
    Velocity v = (l2 + l1) / t;

    // Выводим величины
    cout << l1 / t << endl;
    cout << l2 / l1 / t << endl;
    cout << v << endl;

    // Это не скомпилируется с ошибкой вида
    //    'нельзя из длины вычитать скорость, размерности не сходятся'
    // Mass m = l1 - v;

    return 0;
}
```

После выполнения приведённого main-а на экране будет:

```
5 m^(1)s^(-1)
0.1 s^(-1)
15 m^(1)s^(-1)
```

Дополните данную реализацию:

- Определите ускорение с именем Acceleration и соответственной размерностью
- Допишите операторы вычитания и умножения (не забудьте следить за размерностями)
- Для скорости и ускорения определите специализированные операторы вывода, которые будут писать размерность в более удобном виде - как m/s и m/s^2 соответственно

Ваша реализация должна позволить выполнить следующий код:

```
    // Длина
    Length l1 = {100};
    Length l2 = {200};
    // Время
    Time t = {20};
```

```
    // Скорость
    Velocity v = (l2 - l1) / t;
    Acceleration a = v / t;

    // Выводим величины
    cout << v << endl;
    cout << a << endl;
    cout << (l1 + l2) * v / (t * a) << endl;
```

На экране после этого ожидается:

```
5 m/s
0.25 m/s^2
300 m^(1)
```

```
#include<iostream>

                    using namespace std;

                    // Шаблонный класс для размерной величины
                    template<int L, int M, int T>
                    class DimQ
                    {
                        public:
                            double value;

                            DimQ(double value): value(value) {}
                    };

                    // Псевдонимы типов
                    typedef DimQ<1, 0, 0> Length;
                    typedef DimQ<0, 1, 0> Mass;
                    typedef DimQ<0, 0, 1> Time;
                    typedef DimQ<1, 0, -1> Velocity;

                    // Шаблонный оператор изменения знака
                    template<int L, int M, int T>
                    DimQ<L, M, T> operator-(const DimQ<L, M, T>& q)
                    {
                        return DimQ<L, M, T>(-q.value);
                    }

                    // Шаблонный оператор сложения
                    template<int L, int M, int T>
                    DimQ<L, M, T> operator+(const DimQ<L, M, T>& q1, const DimQ<L, M, T>&
                    q2)
                    {
                        return DimQ<L, M, T>(q1.value + q2.value);
                    }

                    // Шаблонный оператор деления
                    template<
                        int L1, int M1, int T1,
                        int L2, int M2, int T2>
                    DimQ<L1-L2, M1-M2, T1-T2> operator/(const DimQ<L1, M1, T1>& q1, const
                    DimQ<L2, M2, T2>& q2)
                    {
                        return DimQ<L1-L2, M1-M2, T1-T2>(q1.value / q2.value);
                    }

                    // Шаблонный оператор вывода размерной величины
                    template<int L, int M, int T>
                    ostream& operator<<(ostream& os, const DimQ<L, M, T>& q)
                    {
```

```
    os << q.value << " ";

    if (L != 0)
        os << "m^(" << L << ")";

    if (M != 0)
        os << "kg^(" << M << ")";

    if (T != 0)
        os << "s^(" << T << ")";

    return os;

}

int main()
{
    // Длина
    Length l1 = {100};
    Length l2 = {200};
    // Время
    Time t = {20};

    // Скорость
    Velocity v = (l2 + l1) / t;

    // Выводим величины
    cout << l1 / t << endl;
    cout << l2 / l1 / t << endl;
    cout << v << endl;

    // Это не скомпилируется с ошибкой вида
    //    'нельзя из длины вычитать скорость, размерности не сходятся'
    // Mass m = l1 - v;

    return 0;
}
```