Assignment 4

一、概念题

1.1、简述Demeter法则的基本思想；过度使用Demeter法则会带来什么问题。

一个类的成员函数除了能访问自身类结构的类直接子结构（本类的数据成员）外，不能以任何方式依赖任何其他类的结构；只应向某个有限集合的中的类对象发送消息。

它会导致需要写很多包装（wrapper）函数来将函数调用传播到组件，这会带来不少的时间和空间开销（函数内几乎没有内容，会频繁的创建和销毁栈帧）。在函数层面，会导致接口的窄化。在类层面，一个比较宽的接口可能需要生成很多辅助函数。

> https://en.wikipedia.org/wiki/Law_of_Demeter

1.2、简述为什么要对操作符重载进行重载；操作符重载会带来什么问题。

简单方便，提高代码的可读性，比起函数调用，使用运算符更加直观，给开发者带来方便。

很多工作都让编译器做了，复杂的重载有时很难理解代码内部逻辑；而且很容易发生内存泄露；有时会产生歧义。

1.3、简述操作符重载的两种形式；这两种形式有什么区别。

作为一个类的非静态成员函数重载或者作为一个全局(友元)函数来重载。

作为全局函数来实现操作符重载时，其参数至少要有一个具有类、结构、枚举以及它们的引用类型的参数，全部为内置类型不需要重载。

二、编程题

2.1、定义一个日期类Date，其实例为`year`年`month`月`day`日；定义一个时间类Time，其实例为`hour`时`minute`分`second`秒；定义一个日期时间类Datetime，其实例为某日/某时。

```
#include <sstream>
#include <string>
class Date {
  friend class Datetime;
 private:
  int year, month, day;
  static const int dayPerMonth[2][13];
 public:
  Date(int _year, int _month, int _day) {
    year = _year;
    month = _month;
```

```
    day = _day;
  }
  static bool isLeapYear(int year) {
    return year % 400 == 0 || (year % 4 == 0 && year % 100 != 0);
  }
  bool isLeapYear() const {
    return isLeapYear(year);
  }
  bool operator<(const Date &d) const {
    return year < d.year || (year == d.year && (month < d.month ||
        (month == d.month && day < d.day)));
  }
  bool operator==(const Date &d) const {
    return year == d.year && month == d.month && day == d.day;
  }
  Date &operator++() {
    ++day;
    if (day > dayPerMonth[isLeapYear()][month]) {
      day = 1;
      ++month;
      if (month > 12) {
        month = 1;
        ++year;
      }
    }
    return *this;
  }
  Date &operator--() {
    --day;
    if (day <= 0) {
      --month;
      if (month <= 0) {
        --year;
        month = 12;
      }
      day = dayPerMonth[isLeapYear()][month];
    }
    return *this;
  }
  Date operator+(long dayNum) const {
    Date result(*this);
    if (dayNum > 0) {
      for (long i = 0; i < dayNum; ++i) {
        ++result;
      }
    } else {
      for (long i = 0; i > dayNum; --i) {
        --result;
      }
```

```
    }
    return result;
  }
  long operator-(const Date &d) const {
    Date temp(d);
    long result = 0;
    while (temp < *this) {
      ++temp;
      ++result;
    }
    while (*this < temp) {
      --temp;
      --result;
    }
    return result;
  }
};
const int Date::dayPerMonth[2][13] =
    {{0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
     {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}};

class Time {
  friend class Datetime;
 private:
  int hour, minute, second;
 public:
  Time(int _hour, int _minute, int _second) {
    hour = _hour;
    minute = _minute;
    second = _second;
  }
  Time add(long addSecond, long &overflowDay) const {
    int newSecond = second + addSecond % 60;
    addSecond /= 60;
    int newMinute = minute + addSecond % 60 + (newSecond >= 60) -
        (newSecond < 0);
    newSecond = (newSecond + 60) % 60;
    addSecond /= 60;
    int newHour = hour + addSecond % 24 + (newMinute >= 60) -
        (newMinute < 0);
    newMinute = (newMinute + 60) % 60;
    addSecond /= 24;
    overflowDay += addSecond + (newHour >= 24) - (newHour < 0);
    newHour = (newHour + 24) % 24;
    return {newHour, newMinute, newSecond};
  }
  long operator-(const Time &t) const {
    return (hour - t.hour) * 3600 + (minute - t.minute) * 60 + (second
        - t.second);
```

```
  }
  bool operator<(const Time &t) const {
    return hour * 3600 + minute * 60 + second < t.hour * 3600 +
        t.minute * 60 + t.second;
  }
  bool operator==(const Time &t) const {
    return hour * 3600 + minute * 60 + second == t.hour * 3600 +
        t.minute * 60 + t.second;
  }
};

class Datetime {
 private:
  Date date;
  Time time;
 public:
  Datetime(const Date &_date, const Time &_time) : date(_date),
                                                   time(_time) {}
  Datetime operator+(long second) const {
    long addDay = 0;
    Time newTime = time.add(second, addDay);
    Date newDate = date + addDay;
    return {newDate, newTime};
  }
  Datetime operator-(long second) const {
    return *this + (-second);
  }
  long operator-(const Datetime &d) const {
    return (date - d.date) * 86400 + (time - d.time);
  }
  Datetime &operator++() {
    long addDay = 0;
    time = time.add(1, addDay);
    date = date + addDay;
    return *this;
  }
  Datetime &operator--() {
    long addDay = 0;
    time = time.add(-1, addDay);
    date = date + addDay;
    return *this;
  }
  bool operator<(const Datetime &d) const {
    return date < d.date || (date == d.date && time < d.time);
  }
  bool operator==(const Datetime &d) const {
    return date == d.date && time == d.time;
  }

```

```
  std::string to_string() const {
    std::stringstream ss;
    ss << date.year << '.' << date.month << '.' << date.day << '/' <<
       time.hour << '.' << time.minute << '.' << time.second;
    return ss.str();
  }
};
```

咳咳：参考鲁大师的实现