Assignment 6

一、概念简答题

1. 在C++中，protected类成员访问控制的作用是什么?

派生类可以访问基类种的protected成员，但其用户实例不能够调用该成员。

2. 请简述派生类对象的初始化和析构顺序，并简述理由，为什么需要按照这个顺 序?

n派生类对象的初始化由基类和派生类共同完成：

·从基类继承的数据成员由基类的构造函数初始化；

·派生类的数据成员由派生类的构造函数初始化。

n当创建派生类的对象时，

·先执行基类的构造函数，再执行派生类构造函数。

·默认情况下，调用基类的默认构造函数，如果要调用基类的非默认构造函数，则必须在派生类构造函数的成员初始化表中指出。

n当派生类对象消亡时，

·先调用本身类的析构函数，执行完后会自动去调用基类的析构函数。

n如果一个类D既有基类B、又有成员对象类M，则

·在创建D类对象时，构造函数的执行次序为：

B->M->D

·当D类的对象消亡时，析构函数的执行次序为：

D->M->B

二、代码编程题

1.

该题不适合使用public的继承方式，最好使用protected的继承方式。如果使用public的继承方，那么对于Square类的实例对象仍然能够使用Rectangle类的public成员函数，就很有可能造成矛盾的冲突。

2.

```
1
2  #include <stdio.h>
3  #include <assert.h>
4  #include <iostream>
5  using namespace std;
6
7  #define COLOR_NONE                    \033[0m
8  #define FONT_COLOR_RED              \033[0;31m
9  #define FONT_COLOR_BLUE             \033[1;34m
10 #define Assert(expr, type)                                          \
11     if ((expr) == 0)                                                \
12     {                                                               \
13         printf("\033[0;31m%s!! \033[0m", Errors_string[type]); \
14         assert(0);                                                  \
15     }
16 #define Minimum 0
17 #define Maxsecond 60
18 #define Maxminite Maxsecond
19 #define Maxhour 24
20 const char Errors_string[3][10] = {"Herror", "Merror", "Serror"};
21 enum Errors_type
22 {
23     Herror,
24     Merror,
25     Serror
26 };
27
28 class Time
29 {
30 protected:
31     int t_hour, t_minite, t_second;
32
33     void simp()
34     {
35         while (t_second >= Maxsecond)
36         {
37             t_second -= Maxsecond;
38             t_minite++;
39         }
40         while (t_minite >= Maxminite)
41         {
42             t_minite -= Maxminite;
43             t_hour++;
44         }
45         while (t_hour >= Maxhour)
46         {
47             t_hour -= Maxhour;
```

```
        }
    }

public:
    Time(int h = 0, int m = 0, int s = 0)
    {
        set(h, m, s);
    }
    void set(int h, int m, int s)
    {
        Assert(h >= Minimum && h < Maxhour, Herror);
        Assert(m >= Minimum && m < Maxminite, Merror);
        Assert(s >= Minimum && s < Maxsecond, Serror);

        t_hour = h;
        t_minite = m;
        t_second = s;
    }
    void increment()
    {
        t_second++;
        simp();
    }
    void display() const
    {
        printf("Time:  %02d:%02d:%02d\n", t_hour, t_minite,
t_second);
    }

    bool equal(const Time &other_time) const
    {
        return (t_hour == other_time.t_hour && t_minite ==
other_time.t_minite && t_second == other_time.t_second);
    }

    bool less_then(const Time &other_time) const
    {
        return (t_hour < other_time.t_hour) ||
               (t_hour == other_time.t_hour &&
                t_minite < other_time.t_minite) ||
               (t_hour == other_time.t_hour &&
                t_minite == other_time.t_minite &&
                t_second < other_time.t_second);
    }
};
enum TimeZone
{
    W12 = -12,
    W11,
```

```
    W10,
    W9,
    W8,
    W7,
    W6,
    W5,
    W4,
    W3,
    W2,
    W1,
    GMT,
    E1,
    E2,
    E3,
    E4,
    E5,
    E6,
    E7,
    E8,
    E9,
    E10,
    E11,
    E12
};
class ExtTime : public Time
{
public:
    ExtTime() { tz = GMT; }
    ExtTime(int h, int m, int s, TimeZone t) : Time(h, m, s) { tz = t; }
    void set(int h, int m, int s, TimeZone t)
    {
        Time::set(h, m, s);
        tz = t;
    }
    void display() const
    {
        if (tz < 0)
        {
            cout << "西" << -int(tz) << "区\t";
        }
        else if (tz == 0)
        {
            cout<<"格林威治标准时间 (GMT) \t";
        }
        else
        {
            cout <<"东"<<int(tz)<<"区\t";
        }
```

```
143         Time::display();
144     }
145     bool equal(const ExtTime &other_time)const
146     {
147         Time t((t_hour+other_time.tz-tz+24)%24,t_minite,t_second);
148         return t.equal(other_time);
149     }
150     bool less_than(const ExtTime &other_time) const
151     {
152         Time t((t_hour+other_time.tz-tz+24)%24,t_minite,t_second);
153         return t.less_then(other_time);
154     }
155 private:
156     TimeZone tz;
157 };
158 
159 int main()
160 {
161     ExtTime t(1,1,1,W1);
162     ExtTime q(2,2,2,W1);
163     if (t.less_than(q))
164     {
165         cout << "YEs!"<<endl;
166     }
167     else
168     {
169         cout <<"fault"<<endl;
170     }
171 }
172 
```