概念题

1. C++中 Lambda 表达式的语法形式是怎样的？有什么优点？
   [<环境变量使用说明>]<形式参数><返回值类型指定><函数体>
   利用它可以实现把函数的定义和使用合并。对于一些临时使用的简单函数，不用先给出定义并为之命名再通过这个函数的名字来使用它们，给程序编写带来方便。
2. 与正常指针相比，"智能指针"有哪些特点？
   通过该指针对象去访问所指向的对象的成员前能做一些额外的事情，如获得访问次数。

编程题

1. 使用函数对象设计程序获得斐波那契数列并验证其正确性。

```
#include <iostream>
using namespace std;

class fib {
    int starter;
public:
    fib(int i) { starter = i; };
    int operator()() {
        starter += 1;
        if (starter == 2 || starter == 3)
            return 1;
        else
            return fib(starter - 2)() + fib(starter - 3)();
    }
};

int main()
{
    fib Fib1(1), Fib2(2), Fib10(10);
    for(int i = 1; i <= 7; i++)
        cout << Fib1() << " ";
    cout << endl;
    for (int i = 1; i <= 7; i++)
        cout << Fib2() << " ";
    cout << endl;
    for (int i = 1; i <= 7; i++)
        cout << Fib10() << " ";
    cout << endl;
    return 0;
}
```

2. 判断给定的两个点是否一个在圆外一个在圆内。对代码填空使其完成功能:

```
#include <iostream>
using namespace std;

struct Point{
    int x;
    int y;
};

bool IsTrue(Point& rstCenter, int iRadius, Point& P1, Point& P2) {
    auto PointInCircle = [=](Point& p)->bool {return (p.x - rstCenter.x) * (p.x -
rstCenter.x)
        + (p.y - rstCenter.y) * (p.y - rstCenter.y) < iRadius * iRadius; };
    return PointInCircle(P1) != PointInCircle(P2);
}

int main() {
    Point Center, P1, P2;
    Center.x = 0, Center.y = 0;
    int radius = 2;
    P1.x = 1, P1.y = 1;
    P2.x = 3, P2.y = 3;
    cout << IsTrue(Center, radius, P1, P2);
    return 0;
}
```

3. 客户端 B 的日志记录最近一次访问时间。运用智能指针类的知识实现 B:

```
#include <iostream>
#include <time.h>
#include <string>
#pragma warning(disable:4996)
using namespace std;

class A {
public:
    void f1() {}
    void f2() {}
    void f3() {}
    void f4() {}
};

class B {
```

```
    A *p_a;
    string last_time;
public:
    B(){
        p_a = new A;
        last_time = "NULL";
    }
    A* operator->() {
        time_t now_time = time(NULL);
        last_time = asctime(localtime(&now_time));
        return p_a;
    }
    string string_access() const{
        return last_time;
    }
};

void visit(B &b) {
    b->f1();
    b->f2();
    b->f3();
    b->f4();
}

int main()
{
    B b;
    visit(b);
    cout << b.string_access() << endl;
    return 0;
}
```