Assignment submitted on 2020-12-11, 10:31 IST

Your last recorded submission was :

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

class Rectangle {
    int w, h;
public:
    Rectangle(int _w, int _h);
    int area() { return w * h; }
};

class Printer {
    int ar;
public:
    Printer(int _ar);
    void print() { cout << ar; }
};

class PrintableRect : public Rectangle, public Printer {
public:
    PrintableRect(int _w, int _h);
};
Rectangle::Rectangle(int _w, int _h) : w(_w), h(_h) // LINE-1

{ /* do not write anything */ }

Printer::Printer(int _ar) : ar(_ar) // LINE-2

{ /* do not write anything */ }

PrintableRect::PrintableRect(int _w, int _h) : Rectangle( _w, _h), Printer(_w*_h) // LINE-3

{ /* do not write anything */ }

int main() {
    int a, b;

    cin >> a >> b;

    PrintableRect pr(a, b);
    pr.print();

    return 0;
}
```

Assignment submitted on 2020-12-11, 10:32 IST

Your last recorded submission was :

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

int add(int n1, int n2) {
    return n1 + n2;
}

int sub(int n1, int n2) {
    return n1 - n2;
}

int multi(int n1, int n2) {
    return n1 * n2;
}

int divi(int n1, int n2) {
    return n1 / n2;
}
int operation(int data1, int data2, int(*functocall)(int, int)) {

    return (*functocall)(data1,data2); // LINE-1

}

int main() {
    int  a, b, c, d;

    cin >> a >> b >> c;

    int(*fp[])(int,int) = { &add, &sub, &multi, &divi }; // LINE-2
    d = operation(a, b, fp[c]);
    cout << d;

    return 0;
}
```

Assignment submitted on 2020-12-11, 10:51 IST

Your last recorded submission was :

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

class Complex {
private:
    double re, im;
public:
    Complex() : re(0.0), im(0.0) { }

    Complex(double _re, double _im) : re(_re), im(_im) { }

    ~Complex() {}
    double operator ()(){ return sqrt(re * re + im *im); }     // LINE-1

    Complex operator+ (Complex& c) {                          // LINE-2

        Complex t;
        t.re = this->re + c.re;
        t.im = this->im + c.im;
        return t;
    }

    friend ostream& operator<<(ostream&, const Complex&);    // LINE-3

};

ostream& operator<<(ostream &output, const Complex &c) { // LINE-4

    output << c.re << "+j" << c.im;
    return output;
}

int main() {
    int a, b, c, d;
    cin >> a >> b >> c >> d;

    Complex c1(a, b);
    Complex c2(c, d);
    Complex c3 = c1 + c2;

    cout << c3 << " ";
    cout << c3();

    return 0;
}
```

Assignment submitted on 2020-12-11, 10:36 IST

Your last recorded submission was :

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

class loan_manager;    // LINE-1

class customer {
    int _custID;
    string _custName;
    int _cibilScore;
public:
    customer(int custID, string custName, int cibilScore) : _custID(custID),
        _custName(custName), _cibilScore(cibilScore){}
    void changeScore(int change) const {    // LINE-2

        const_cast<customer*>(this)->_cibilScore += change;    // LINE-3

    }
    friend class loan_manager;    // LINE-4
};

class loan_manager {
    int _mgrID;
public:
    loan_manager(int mgrID) : _mgrID(mgrID){}
    void evaluteCustomer(const customer& c){
        cout << _mgrID << " : " << c._custID << " : " << c._custName <<
            " : " << c._cibilScore << endl;
    }
};

int main() {
    int cid, cibil, mid, change;
    string name;

    cin >> cid >> name >> cibil >> change >> mid;

    const customer c(cid, name, cibil);
    c.changeScore(change);

    loan_manager l(mid);
    l.evaluteCustomer(c);

    return 0;
}
```

Assignment submitted on 2020-12-11, 20:10 IST

Your last recorded submission was :

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

class GeoTransformation;                                        // LINE-1

class Point { int _x, _y;
public:
    Point(int x = 0, int y = 0) : _x(x), _y(y) { }
    Point(const Point &tp): _x(tp._x), _y(tp._y) { }            // LINE-2

    friend ostream & operator<<(ostream&, const Point&);    // LINE-3

   friend GeoTransformation;                                    // LINE-4
};

ostream& operator<<(ostream &out, const Point &p) { // LINE-5
    out << "(" << p._x << ", " << p._y << ")" << endl;
    return out;
}

class GeoTransformation { Point p;
public:
    GeoTransformation(Point& pt) : p(pt) { }
    Point operator+(int t);
};

Point GeoTransformation::operator+(int t) {
    Point tp;
    tp._x = p._x + t;
    tp._y = p._y + t;
    return tp;
}

int main() {
    int i, j, k;

    cin >> i >> j >> k;

    Point p1(i, j);
    GeoTransformation g(p1);

    cout << g + k;

    return 0;
}
```

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

class Customer{
private:
    int _custID;
    string _name;

    string _custCity;   // LINE-1

public:
    Customer(int custID, string name, string custCity) :
        _custID(custID), _name(name), _custCity(custCity) { }

    Customer(const Customer &_customer) : // LINE-2

        _custID(_customer._custID * 100), _name(_customer._name),
        _custCity(_customer._custCity) { }

    void changeCity(string custCity) { _custCity = custCity; } // LINE-3

   void printCustomer(); // LINE-4
};

class Invoice {
private:
    int _invoiceID;
    Customer _customer;
    double _payable;
public:
    Invoice(int invoiceID, const Customer customer, double payable) :
        _invoiceID(invoiceID), _customer(customer), _payable(payable){}
    void printInvoice();
};

void Customer::printCustomer() { // LINE-5
    cout << _custID << " : " << _name << "[" << _custCity << "]" << " - ";
}

void Invoice::printInvoice() {
    _customer.printCustomer();
    cout << _invoiceID << " : " << _payable << endl;
}

int main() {
    int i, j;
    double k;
    string n, c, c1;

    cin >> i >> n >> c >> c1 >> j >> k;

    Customer ct(i, n, c);
    ct.changeCity(c1);

    Invoice in(j, ct, k);
    in.printInvoice();

    return 0;
}
```

```cpp
#include <iostream>
#include <string>
#include <typeinfo>
using namespace std;

class Playable {
public:
  virtual void play() = 0;    // LINE-1
};

class Music : public Playable {
public:
    string getMusicType(int i) {      // LINE-2

        string MusicType[] = { "Rock", "Jazz", "Pop", "Folk",
                               "Classical", "Blues", "Heavy Metal" };
        if (i >= 0 && i < 7)
            return MusicType[i];
        else
            return "unknown";
    }
    void play();
};

class Game : public Playable {
public:
    string getGameType(int i) {      // LINE-3

        string GameType[] = { "Action", "Action-adventure", "Adventure",
                              "Role-playing", "Simulation", "Strategy",
                              "Sports", "Puzzle" };
        if (i >= 0 && i < 8)
            return GameType[i];
        else
            return "unknown";
    }
    void play();
};

class Player {
    Playable *_pt;
    string type;
public:
    Player(Playable *pt) : _pt(pt) { }
    void play(int i){
        _pt->play();
        if (typeid(*_pt) == typeid(Music))      // LINE-4

            type = ((Music*)_pt)->getMusicType(i);

        if (typeid(*_pt) == typeid(Game))       // LINE-5


            type = ((Game*)_pt)->getGameType(i);
        cout << type << endl;
    }
};

void Music::play(){ cout << "play music : "; }
void Game::play(){ cout << "play game : "; }

int main() {
    int i, j;
    cin >> i >> j;

    Game g;
    Music m;
    Player p1(&g);
    Player p2(&m);

    p1.play(i);
    p2.play(j);

    return 0;
}
```

Assignment submitted on 2020-12-11, 20:24 IST

Your last recorded submission was :

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

template<class T>                               // LINE-1

class MySequence {
    vector<T> _arr;
    const int _len;
public:
    MySequence(int len = 0) : _len(len) {

        _arr.resize(_len);                      // LINE-2

    }
    T& operator [] (int i) {                // LINE-3

        if (0 <= i && i < _len)
            return _arr[i];

        throw 5;                        // LINE-4
    }
};
int main() {
    try {
        MySequence<int> s1(5);

        int i, i1, i2;
        cin >> i1 >> i2;

        for (i = i1; i <= i2; i++)
            s1[i] = i * 10;

        for (i = i1; i <= i2; i++)
            cout << s1[i] << " ";

        MySequence<char> s2(5);
        for (i = i1; i <= i2; i++)
            s2[i] = 65 + i;

        for (i = i1; i <= i2; i++)
            cout << s2[i] << " ";
    }
    catch (int i) {
        cout << "error:" << i;
    }

    return 0;
}
```

Assignment submitted on 2020-12-11, 20:19 IST

Your last recorded submission was :

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

class ComplexNum {
    double _r, _i;
public:
    ComplexNum(double r=0, double i=0) : _r(r), _i(i) { }         // LINE-1

    ComplexNum(const ComplexNum& c) : _r(c._r), _i(c._i) { }   // LINE-2

    ComplexNum operator=(const ComplexNum& c){                          // LINE-3
        _r = c._r;
        _i = c._i;
        return *this;
    }

    ComplexNum operator*(const ComplexNum& c){
        ComplexNum tc;
        tc._r = (_r * c._r) - (_i * c._i);
        tc._i = (_r * c._i) + (_i * c._r);
        return tc;
    }

    void print() {
        cout << _r << " + " << _i << "i";
    }
};

int main() {
    double r1, i1, r2, i2;
    cin >> r1 >> i1 >> r2 >> i2;

    ComplexNum c1(r1, i1);
    ComplexNum c2(r2, i2);

    ComplexNum c3 = c1;

    c1 = c2;

    ComplexNum c4 = c1 * c3;
    c4.print();

    return 0;
}
```

Assignment submitted on 2020-12-11, 20:11 IST

Your last recorded submission was :

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

class ShapeAttribute {

protected:
    double _s;

public:
    ShapeAttribute(double s) : _s(s) { }

    virtual double getVal() = 0;
};
class Volume : public ShapeAttribute {                    // LINE-1
public:
    Volume(double s) : ShapeAttribute(s) { }
    double getVal() {
        return _s * _s * _s;
    }
};

class Area : public ShapeAttribute {                     // LINE-2
public:
    Area(double s) : ShapeAttribute(s) { }
    double getVal(){
        return 6 * _s * _s;
    }
};

class Cube : public Volume, public Area {                    // LINE-3
public:
    Cube(double s) : Volume(s), Area(s){}

    double getArea() { return Area::getVal(); }   // LINE-4

    double getVolume() { return Volume::getVal(); } // LINE-5
};
int main() {
    int i;
    cin >> i;

    Cube c(i);
    cout << c.getArea() << " " << c.getVolume();

    return 0;
}
```

Assignment submitted on 2020-12-11, 20:57 IST

Your last recorded submission was :

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

class Celsius;      // LINE-1

class Fahrenheit {
    double _temp;
public:
    Fahrenheit(double temp = 0.0) : _temp(temp) { }

    friend Celsius operator+(Celsius&,Fahrenheit&);     // LINE-2

    operator Celsius()const;     // LINE-3
};

class Celsius {
    double _temp;
public:
    Celsius(double temp = 0.0) : _temp(temp) { }
    void show();

    friend Celsius operator+(Celsius&,Fahrenheit&);       // LINE-4

    friend Fahrenheit :: operator Celsius()const;       // LINE-5
};

Fahrenheit::operator Celsius() const {     // LINE-6

    Celsius c;
    c._temp = (_temp - 32) * ((double)5 / 9);
    return c;
}
Celsius operator+(Celsius& t1, Fahrenheit& t2) {
    Celsius t;
    t._temp = t1._temp + ((Celsius)t2)._temp;
    return t;
}

void Celsius::show() { cout << _temp << "C" << endl; }
int main() {
    double t1, t2;
    cin >> t1 >> t2;

    Celsius c(t1);
    Fahrenheit f(t2);

    Celsius t = c + f;
    t.show();

    return 0;
}
```