

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

Your last recorded submission was :

```
1 #include <iostream>
2 using namespace std;
3
4 class Rectangle {
5     int w, h;
6 public:
7     Rectangle(int _w, int _h);
8     int area() { return w * h; }
9 };
10
11 class Printer {
12     int ar;
13 public:
14     Printer(int _ar);
15     void print() { cout << ar; }
16 };
17
18 class PrintableRect : public Rectangle, public Printer {
19 public:
20     PrintableRect(int _w, int _h);
21 };
22 Rectangle::Rectangle(int _w, int _h) : w(_w), h(_h) // LINE-1
23 { /* do not write anything */ }
24
25 Printer::Printer(int _ar) : ar(_ar) // LINE-2
26 { /* do not write anything */ }
27
28 PrintableRect::PrintableRect(int _w, int _h) : Rectangle(_w, _h), Printer(_w*_h) // LINE-3
29 { /* do not write anything */ }
30
31
32
33
34 int main() {
35     int a, b;
36
37     cin >> a >> b;
38
39     PrintableRect pr(a, b);
40     pr.print();
41
42     return 0;
43 }
```

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

Your last recorded submission was :

```
1 #include <iostream>
2 using namespace std;
3
4 int add(int n1, int n2) {
5     return n1 + n2;
6 }
7
8 int sub(int n1, int n2) {
9     return n1 - n2;
10 }
11
12 int multi(int n1, int n2) {
13     return n1 * n2;
14 }
15
16 int divi(int n1, int n2) {
17     return n1 / n2;
18 }
19 int operation(int data1, int data2, int(*functocall)(int, int)) {
20
21     return (*functocall)(data1,data2); // LINE-1
22
23 }
24
25 int main() {
26     int a, b, c, d;
27
28     cin >> a >> b >> c;
29
30     int(*fp[])(int,int) = { &add, &sub, &multi, &divi }; // LINE-2
31     d = operation(a, b, fp[c]);
32     cout << d;
33
34     return 0;
35 }
```

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

Your last recorded submission was :

```
1 #include <iostream>
2 #include <cmath>
3 using namespace std;
4
5 class Complex {
6 private:
7     double re, im;
8 public:
9     Complex() : re(0.0), im(0.0) { }
10
11     Complex(double _re, double _im) : re(_re), im(_im) { }
12
13     ~Complex() {}
14     double operator ()(){ return sqrt(re * re + im * im); } // LINE-1
15
16     Complex operator+ (Complex& c) { // LINE-2
17
18         Complex t;
19         t.re = this->re + c.re;
20         t.im = this->im + c.im;
21         return t;
22     }
23
24     friend ostream& operator<<(ostream&, const Complex&); // LINE-3
25 };
26
27 ostream& operator<<(ostream &output, const Complex &c) { // LINE-4
28
29     output << c.re << "+" << c.im;
30     return output;
31 }
32
33 int main() {
34     int a, b, c, d;
35     cin >> a >> b >> c >> d;
36
37     Complex c1(a, b);
38     Complex c2(c, d);
39     Complex c3 = c1 + c2;
40
41     cout << c3 << " ";
42     cout << c3();
43
44     return 0;
45 }
46
```

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

Your last recorded submission was :

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 class loan_manager;    // LINE-1
6
7 class customer {
8     int _custID;
9     string _custName;
10    int _cibilScore;
11 public:
12     customer(int custID, string custName, int cibilScore) : _custID(custID),
13         _custName(custName), _cibilScore(cibilScore){}
14     void changeScore(int change) const {    // LINE-2
15
16         const_cast<customer*>(this)->_cibilScore += change;    // LINE-3
17     }
18     friend class loan_manager;    // LINE-4
19 };
20
21 class loan_manager {
22     int _mgrID;
23 public:
24     loan_manager(int mgrID) : _mgrID(mgrID){}
25     void evaluateCustomer(const customer& c){
26         cout << _mgrID << " : " << c._custID << " : " << c._custName <<
27             " : " << c._cibilScore << endl;
28     }
29 };
30
31 int main() {
32     int cid, cibil, mid, change;
33     string name;
34
35     cin >> cid >> name >> cibil >> change >> mid;
36
37     const customer c(cid, name, cibil);
38     c.changeScore(change);
39
40     loan_manager l(mid);
41     l.evaluateCustomer(c);
42
43     return 0;
44 }
45 }
```

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

Your last recorded submission was :

```
1 #include <iostream>
2 using namespace std;
3
4 class GeoTransformation;           // LINE-1
5
6 class Point { int _x, _y;
7 public:
8     Point(int x = 0, int y = 0) : _x(x), _y(y) { }
9     Point(const Point &tp): _x(tp._x), _y(tp._y) { }           // LINE-2
10
11     friend ostream & operator<<(ostream&, const Point&);      // LINE-3
12
13     friend GeoTransformation;           // LINE-4
14 };
15
16 ostream& operator<<(ostream &out, const Point &p) { // LINE-5
17     out << "(" << p._x << ", " << p._y << ")" << endl;
18     return out;
19 }
20
21 class GeoTransformation { Point p;
22 public:
23     GeoTransformation(Point& pt) : p(pt) { }
24     Point operator+(int t);
25 };
26
27 Point GeoTransformation::operator+(int t) {
28     Point tp;
29     tp._x = p._x + t;
30     tp._y = p._y + t;
31     return tp;
32 }
33
34 int main() {
35     int i, j, k;
36
37     cin >> i >> j >> k;
38
39     Point p1(i, j);
40     GeoTransformation g(p1);
41
42     cout << g + k;
43
44     return 0;
45 }
```

```

1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 class Customer{
6 private:
7     int _custID;
8     string _name;
9
10    string _custCity;    // LINE-1
11
12 public:
13     Customer(int custID, string name, string custCity) :
14         _custID(custID), _name(name), _custCity(custCity) { }
15
16     Customer(const Customer &_customer) : // LINE-2
17         _custID(_customer._custID * 100), _name(_customer._name),
18         _custCity(_customer._custCity) { }
19
20     void changeCity(string custCity) { _custCity = custCity; } // LINE-3
21
22     void printCustomer(); // LINE-4
23 };
24
25 class Invoice {
26 private:
27     int _invoiceID;
28     Customer _customer;
29     double _payable;
30 public:
31     Invoice(int invoiceID, const Customer customer, double payable) :
32         _invoiceID(invoiceID), _customer(customer), _payable(payable){}
33     void printInvoice();
34 };
35
36 void Customer::printCustomer() { // LINE-5
37     cout << _custID << " : " << _name << "[" << _custCity << "]" << " - ";
38 }
39
40 void Invoice::printInvoice() {
41     _customer.printCustomer();
42     cout << _invoiceID << " : " << _payable << endl;
43 }
44
45 int main() {
46     int i, j;
47     double k;
48     string n, c, c1;
49
50     cin >> i >> n >> c >> c1 >> j >> k;
51
52     Customer ct(i, n, c);
53     ct.changeCity(c1);
54
55     Invoice in(j, ct, k);
56     in.printInvoice();
57
58     return 0;
59 }
60

```



```

1 #include <iostream>
2 #include <string>
3 #include <typeinfo>
4 using namespace std;
5
6 class Playable {
7 public:
8     virtual void play() = 0;    // LINE-1
9 };
10
11 class Music : public Playable {
12 public:
13     string getMusicType(int i) {    // LINE-2
14
15         string MusicType[] = { "Rock", "Jazz", "Pop", "Folk",
16                                 "Classical", "Blues", "Heavy Metal" };
17         if (i >= 0 && i < 7)
18             return MusicType[i];
19         else
20             return "unknown";
21     }
22     void play();
23 };
24
25 class Game : public Playable {
26 public:
27     string getGameType(int i) {    // LINE-3
28
29         string GameType[] = { "Action", "Action-adventure", "Adventure",
30                                "Role-playing", "Simulation", "Strategy",
31                                "Sports", "Puzzle" };
32         if (i >= 0 && i < 8)
33             return GameType[i];
34         else
35             return "unknown";
36     }
37     void play();
38 };
39
40 class Player {
41     Playable *_pt;
42     string type;
43 public:
44     Player(Playable *_pt) : _pt(pt) { }
45     void play(int i){
46         _pt->play();
47         if (typeid(*_pt) == typeid(Music))    // LINE-4
48             type = ((Music*)_pt)->getMusicType(i);
49         if (typeid(*_pt) == typeid(Game))    // LINE-5
50             type = ((Game*)_pt)->getGameType(i);
51         cout << type << endl;
52     }
53 };
54
55 void Music::play(){ cout << "play music : "; }
56 void Game::play(){ cout << "play game : "; }
57
58 int main() {
59     int i, j;
60     cin >> i >> j;
61
62     Game g;
63     Music m;
64     Player p1(&g);
65     Player p2(&m);
66
67     p1.play(i);
68     p2.play(j);
69
70     return 0;
71 }

```

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

Your last recorded submission was :

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 template<class T>           // LINE-1
6
7 class MySequence {
8     vector<T> _arr;
9     const int _len;
10 public:
11     MySequence(int len = 0) : _len(len) {
12
13         _arr.resize(_len);           // LINE-2
14
15     }
16     T& operator [] (int i) {         // LINE-3
17
18         if (0 <= i && i < _len)
19             return _arr[i];
20
21         throw 5;                     // LINE-4
22     }
23 };
24 int main() {
25     try {
26         MySequence<int> s1(5);
27
28         int i, i1, i2;
29         cin >> i1 >> i2;
30
31         for (i = i1; i <= i2; i++)
32             s1[i] = i * 10;
33
34         for (i = i1; i <= i2; i++)
35             cout << s1[i] << " ";
36
37         MySequence<char> s2(5);
38         for (i = i1; i <= i2; i++)
39             s2[i] = 65 + i;
40
41         for (i = i1; i <= i2; i++)
42             cout << s2[i] << " ";
43     }
44     catch (int i) {
45         cout << "error:" << i;
46     }
47
48     return 0;
49 }
```



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

Your last recorded submission was :

```
1  #include <iostream>
2  using namespace std;
3
4  class ComplexNum {
5      double _r, _i;
6  public:
7      ComplexNum(double r=0, double i=0) : _r(r), _i(i) { }           // LINE-1
8
9      ComplexNum(const ComplexNum& c) : _r(c._r), _i(c._i) { }       // LINE-2
10
11     ComplexNum operator=(const ComplexNum& c){                      // LINE-3
12         _r = c._r;
13         _i = c._i;
14         return *this;
15     }
16
17     ComplexNum operator*(const ComplexNum& c){
18         ComplexNum tc;
19         tc._r = (_r * c._r) - (_i * c._i);
20         tc._i = (_r * c._i) + (_i * c._r);
21         return tc;
22     }
23
24     void print() {
25         cout << _r << " + " << _i << "i";
26     }
27 };
28
29 int main() {
30     double r1, i1, r2, i2;
31     cin >> r1 >> i1 >> r2 >> i2;
32
33     ComplexNum c1(r1, i1);
34     ComplexNum c2(r2, i2);
35
36     ComplexNum c3 = c1;
37
38     c1 = c2;
39
40     ComplexNum c4 = c1 * c3;
41     c4.print();
42
43     return 0;
44 }
```

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

Your last recorded submission was :

```
1 #include <iostream>
2 using namespace std;
3
4 class ShapeAttribute {
5
6 protected:
7     double _s;
8
9 public:
10     ShapeAttribute(double s) : _s(s) { }
11
12     virtual double getVal() = 0;
13 };
14 class Volume : public ShapeAttribute {           // LINE-1
15 public:
16     Volume(double s) : ShapeAttribute(s) { }
17     double getVal() {
18         return _s * _s * _s;
19     }
20 };
21
22 class Area : public ShapeAttribute {             // LINE-2
23 public:
24     Area(double s) : ShapeAttribute(s) { }
25     double getVal(){
26         return 6 * _s * _s;
27     }
28 };
29
30 class Cube : public Volume, public Area {         // LINE-3
31 public:
32     Cube(double s) : Volume(s), Area(s){}
33
34     double getArea() { return Area::getVal(); }   // LINE-4
35
36     double getVolume() { return Volume::getVal(); } // LINE-5
37 };
38 int main() {
39     int i;
40     cin >> i;
41
42     Cube c(i);
43     cout << c.getArea() << " " << c.getVolume();
44
45     return 0;
46 }
```

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

Your last recorded submission was :

```
1  #include <iostream>
2  using namespace std;
3
4  class Celsius;      // LINE-1
5
6  class Fahrenheit {
7      double _temp;
8  public:
9      Fahrenheit(double temp = 0.0) : _temp(temp) { }
10
11     friend Celsius operator+(Celsius&,Fahrenheit&);    // LINE-2
12
13     operator Celsius()const;        // LINE-3
14 };
15
16 class Celsius {
17     double _temp;
18 public:
19     Celsius(double temp = 0.0) : _temp(temp) { }
20     void show();
21
22     friend Celsius operator+(Celsius&,Fahrenheit&);    // LINE-4
23
24     friend Fahrenheit :: operator Celsius()const;    // LINE-5
25 };
26
27 Fahrenheit::operator Celsius() const {    // LINE-6
28
29     Celsius c;
30     c._temp = (_temp - 32) * ((double)5 / 9);
31     return c;
32 }
33 Celsius operator+(Celsius& t1, Fahrenheit& t2) {
34     Celsius t;
35     t._temp = t1._temp + ((Celsius)t2)._temp;
36     return t;
37 }
38
39 void Celsius::show() { cout << _temp << "C" << endl; }
40 int main() {
41     double t1, t2;
42     cin >> t1 >> t2;
43
44     Celsius c(t1);
45     Fahrenheit f(t2);
46
47     Celsius t = c + f;
48     t.show();
49
50     return 0;
51 }
```