NAME : PATOLIYA MAULIKKUMAR NARENDRABHAI

BRANCH : CE (24CEUOG121)

BATCH : A2

ROLL NO. : 041

SUB : ADVANCED C++ PROGRAMMING CONCEPTS

FOCUS: OVERLOAD RESOLUTION | TYPE CONVERSION

Q-1

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
    int rs;
    int ps;
public:

 INR():rs(0),ps(0){}

 INR(float r){
    rs = r;
    ps = round((r-rs)*100);
 }

INR(int r , int p){
    rs = r + p/100 ;
    ps = p%100;
 }

 operator float(){
    return (rs + ps/100.0);
 }

 INR operator+( INR io2){
    return INR( float(*this) + float(io2) );
 }

 void print(){
    cout << fixed << "INR = " << rs << "."
     << ps << endl;
 }

};
```

```
class USD{
    int dlr;
    int cnt;

 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
    dlr = d;
    cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
    dlr = d + c/100;
    cnt = c%100;
 }

 USD(INR i){
    USD(i/exchange);
 }

 operator float(){
    return ( dlr + cnt/100.0);
 }

 operator INR(){
    return INR( float(*this) * exchange);
 }

 USD operator+( USD uo2){
    return USD( float(*this) + float(uo2) );
 }

 void print(){
    cout << fixed << "USD = " << dlr << "."
    << cnt << endl;
 }

};
```

```
int main(){

  INR io1(358.5);
  USD uo1(38.538);

  cout << "io1 : ";
  io1.print();
  cout << "uo1 : ";
  uo1.print();

  USD uo2(io1);
  cout << "uo2 : ";
  uo2.print();

  return 0;
}
```

output ?

Maulik N. Patoliya

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator+( INR io2){
 return INR( float(*this) + float(io2) );
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }

};
```

```
class USD{
 int dlr;
 int cnt;

 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 operator INR(){
 return INR( float(*this) * exchange);
 }

 USD operator+( USD uo2){
 return USD( float(*this) + float(uo2) );
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```
int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();


 USD uo2(io1);
 cout << "uo2 : ";
 uo2.print();

 return 0;
}
```

- This creates a temporary USD object but does not initialize dlr and cnt in the current object.
- It does nothing to modify this.
- This results in an uninitialized object, leading to incorrect or undefined behavior.

OUTPUT :

```
 io1 : INR = 358.50
uo1 : USD = 38.54
uo2 : USD = -2.6422280
```

Sol_ - 1

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator+( INR io2){
 return INR( float(*this) + float(io2) );
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }

};
```

```
class USD{
 int dlr;
 int cnt;

 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 operator INR(){
 return INR( float(*this) * exchange);
 }

 USD operator+( USD uo2){
 return USD( float(*this) + float(uo2) );
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```
int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();


 USD uo2(io1);
 cout << "uo2 : ";
 uo2.print();

 return 0;
}
```

OUTPUT :

```
 io1 : INR = 358.50
uo1 : USD = 38.54
uo2 : USD = 4.19
```

Maulik N. Patoliya

Sol_ - 2

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator+( INR io2){
 return INR( float(*this) + float(io2) );
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }

};
```

```
class USD{
 int dlr;
 int cnt;

 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i) : USD(i/exchange){
  // *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 operator INR(){
 return INR( float(*this) * exchange);
 }

 USD operator+( USD uo2){
 return USD( float(*this) + float(uo2) );
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```
int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();


 USD uo2(io1);
 cout << "uo2 : ";
 uo2.print();

 return 0;
}
```

OUTPUT :

```
 io1 : INR = 358.50
uo1 : USD = 38.54
uo2 : USD = 4.19
```

Q-2

```cpp
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator+( INR io2){
 return INR( float(*this) + float(io2) );
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }

};
```

```cpp
class USD{
 int dlr;
 int cnt;

 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 operator INR(){
 return INR( float(*this) * exchange);
 }

 USD operator+( USD uo2){
 return USD( float(*this) + float(uo2) );
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```cpp
INR operator+( INR i , USD u){
 return INR( float(i) + (float(u)*exchange));
}

USD operator+( USD u , INR i ){
 return USD( float(u) + (float(i)/exchange));
}

int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2(io1);
 cout << "uo2 : ";
 uo2.print();

 float x = uo1 + uo2 , y = io1 + io2 ;
 float z = io1 + uo2 , p = uo2 + io1 ;

 cout << fixed << setprecision(2) << "X:"
 << x << " Y:" << y << " z:" << z <<
 " P:" << p << endl;
 return 0;
}
```

output ?

Maulik N. Patoliya

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator+( INR io2){
 return INR( float(*this) + float(io2) );
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }

};
```

```
class USD{
 int dlr;
 int cnt;

 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 operator INR(){
 return INR( float(*this) * exchange);
 }

 USD operator+( USD uo2){
 return USD( float(*this) + float(uo2) );
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```
INR operator+( INR i , USD u){
 return INR( float(i) + (float(u)*exchange));
}

USD operator+( USD u , INR i ){
 return USD( float(u) + (float(i)/exchange));
}

int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 //error : more than one instance of constructor
 "INR::INR" matches the argument list
 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2(io1);
 cout << "uo2 : ";
 uo2.print();
```

```
float x = uo1 + uo2 , y = io1 + io2 ;
 float z = io1 + uo2 , p = uo2 + io1 ;

 cout << fixed << setprecision(2) << "X:"
 << x << " Y:" << y << endl << " z:" << z <<
 " P:" << p << endl;
 return 0;
}
```

Maulik N. Patoliya

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator+( INR io2){
 return INR( float(*this) + float(io2) );
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }

};
```

```
class USD{
 int dlr;
 int cnt;

 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 operator INR(){
 return INR( float(*this) * exchange);
 }

 USD operator+( USD uo2){
 return USD( float(*this) + float(uo2) );
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```
INR operator+( INR i , USD u){
 return INR( float(i) + (float(u)*exchange));
}

USD operator+( USD u , INR i ){
 return USD( float(u) + (float(i)/exchange));
}

int main(){

 INR io1(358.5);
 USD uo1(38.538)

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 // INR io2(uo1);
 INR io2 = uo1 ;
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2(io1);
 cout << "uo2 : ";
 uo2.print();

float x = uo1 + uo2 , y = io1 + io2 ;
 float z = io1 + uo2 , p = uo2 + io1 ;

 cout << fixed << setprecision(2) << "X:"
 << x << " Y:" << y << endl << " z:" << z <<
 " P:" << p << endl;
 return 0;
}
```

Sol_ - 1

OUTPUT :

```
 io1 : INR = 358.50
uo1 : USD = 38.54
 io2 : INR  = 3293.63
 io3 : INR = 3293.63
uo2 : USD = 4.19
X : 42.73    Y : 3652.13
Z : 716.58   P : 8.38
```

Maulik N. Patoliya

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
 public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

 INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 INR(USD u);

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator+( INR io2){
 return INR( float(*this) + float(io2) );
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }
};

class USD{
 int dlr;
 int cnt;
 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 operator INR(){
 return INR( float(*this) * exchange);
 }

 USD operator+( USD uo2){
 return USD( float(*this) + float(uo2) );
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};

INR::INR(USD u){
 *this = INR(float(u)*exchange);
}

INR operator+( INR i , USD u){
 return INR( float(i) + (float(u)*exchange));
}

USD operator+( USD u , INR i ){
 return USD( float(u) + (float(i)/exchange));
}

int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 // io3 = uo1; error : ambiguity
 // 1. Constructor , 2. operator overload

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2 = io1;
 cout << "uo2 : ";
 uo2.print();

 float x = uo1 + uo2 ;
 float y = io1 + io2 ;
 float z = io1 + uo2 ;
 float p = uo2 + io1 ;

cout << fixed << setprecision(2) << "X:"
 << x << " Y:" << y << endl << "z:" <<
 z << " P:" << p << endl;
 return 0;

}
```

Sol_ - 2

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
 public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

 INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 INR(USD u);

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator+( INR io2){
 return INR( float(*this) + float(io2) );
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }
};

class USD{
 int dlr;
 int cnt;
 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 /* operator INR(){
 return INR( float(*this) * exchange);
 } */

 USD operator+( USD uo2){
 return USD( float(*this) + float(uo2) );
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};

INR::INR(USD u){
 *this = INR(float(u)*exchange);
}

INR operator+( INR i , USD u){
 return INR( float(i) + (float(u)*exchange));
}

USD operator+( USD u , INR i ){
 return USD( float(u) + (float(i)/exchange));
}

int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2 = io1;
 cout << "uo2 : ";
 uo2.print();

 float x = uo1 + uo2 ;
 float y = io1 + io2 ;
 float z = io1 + uo2 ;
 float p = uo2 + io1 ;

 cout << fixed << setprecision(2) << "X:"
 << x << " Y:" << y << endl << "z:" <<
 z << " P:" << p << endl;
 return 0;

}
```

Sol_ - 2

OUTPUT :

```
 io1 : INR = 358.50
uo1 : USD = 38.54
 io2 : INR  = 3293.63
 io3 : INR = 3293.63
uo2 : USD = 4.19
X : 42.73    Y : 3652.13
Z : 716.58   P : 8.38
```

Maulik N. Patoliya

Conclusion

- ➯ Here rather than use of operator overload for USD -> INR , constructor is more efficient .

- ➯ Because , with using constructor ;

  INR io2(uo1) ;
  IINR io2 = uo1 ;

  both are work fine

- ➯ But , with using operator overload ;

  INR io2(uo1) ; // Gives ambiguity error
  INR io3 = uo1; // Compile and work fine

Q-2

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
 public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

 INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 INR(USD u);

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator+( INR io2){
 return INR( float(*this) + float(io2) );
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }
};
```

```
class USD{
 int dlr;
 int cnt;
 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }


 USD operator+( USD uo2){
 return USD( float(*this) + float(uo2) );
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```
INR::INR(USD u){
 *this = INR(float(u)*exchange);
}

INR operator+( INR i , USD u){
 return INR( (i) + (float(u)*exchange));
}

USD operator+( USD u , INR i ){
 return USD( (u) + (float(i)/exchange));
}

int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2 = io1;
 cout << "uo2 : ";
 uo2.print();

 float x = uo1 + uo2 ;
 float y = io1 + io2 ;
 float z = io1 + uo2 ;
 float p = uo2 + io1 ;
```

```
cout << fixed << setprecision(2) << "X:"
 << x << " Y:" << y << endl << "z:" <<
 z << " P:" << p << endl;
 return 0;

}
```

output ?

Maulik N. Patoliya

```cpp
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
 public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

 INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 INR(USD u);

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator+( INR io2){
 return INR( float(*this) + float(io2) );
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }
};
```

```cpp
class USD{
 int dlr;
 int cnt;
 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }


 USD operator+( USD uo2){
 return USD( float(*this) + float(uo2) );
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```cpp
INR::INR(USD u){
 *this = INR(float(u)*exchange);
}

// error : ambiguity (more than one operator "+" matches these operands:)

INR operator+( INR i , USD u){
 return INR( (i) + (float(u)*exchange));
}

USD operator+( USD u , INR i ){
 return USD( (u) + (float(i)/exchange));
}

int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2 = io1;
 cout << "uo2 : ";
 uo2.print();

 float x = uo1 + uo2 ;
 float y = io1 + io2 ;
 float z = io1 + uo2 ;
 float p = uo2 + io1 ;
```

```cpp
cout << fixed << setprecision(2) << "X:"
 << x << " Y:" << y << endl << "z:" <<
 z << " P:" << p << endl;
 return 0;

}
```

Maulik N. Patoliya

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
 public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

 INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 INR(USD u);

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator+( INR io2){
 return INR( float(*this) + float(io2) );
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }
};
```

```
class USD{
 int dlr;
 int cnt;
 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }


 USD operator+( USD uo2){
 return USD( float(*this) + float(uo2) );
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```
INR::INR(USD u){
 *this = INR(float(u)*exchange);
}

INR operator+( INR i , USD u){
 return INR( float (i) + (float(u)*exchange));
}

USD operator+( USD u , INR i ){
 return USD( float (u) + (float(i)/exchange));
}

int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2 = io1;
 cout << "uo2 : ";
 uo2.print();

 float x = uo1 + uo2 ;
 float y = io1 + io2 ;
 float z = io1 + uo2 ;
 float p = uo2 + io1 ;
```

```
 cout << fixed << setprecision(2) << "X:"
  << x << " Y:" << y << endl << "z:" <<
  z << " P:" << p << endl;
  return 0;

 }
```

OUTPUT :

```
 io1 : INR = 358.50
uo1 : USD = 38.54
 io2 : INR  = 3293.63
 io3 : INR = 3293.63
uo2 : USD = 4.19
X : 42.73     Y : 3652.13
Z : 716.58   P : 8.38
```

Maulik N. Patoliya

Why does ambiguity occur ?

The problem happens because of conflicting implicit conversions between INR, USD, and float:

1. Implicit float Conversions :
   - INR and USD both have an operator float(), allowing them to be automatically converted to float.
   - When performing i + float(u) * exchange, the compiler must decide how to convert i to float, which causes ambiguity.

2. Implicit Constructors INR(USD) and USD(INR) :
   - Since INR(USD u) and USD(INR i) exist, INR and USD can be directly converted into one another.
   - The compiler sees multiple valid ways to evaluate INR((i) + (float(u)*exchange)):
     - Convert INR i to float first, add, and construct INR.
     - Convert USD u to INR directly via INR(USD), then add.
   - The compiler gets confused because both conversions are valid.

3. Compiler’s Confusion in operator+ :

- The expression return INR((i) + (float(u)*exchange)); contains:
  - i → float (via operator float())
  - u → float (via operator float())
- The compiler doesn't know which conversion should take precedence:
  - INR i → float, then float + float, then INR(float)
  - OR USD u → INR, then INR + INR

⇨ Two valid interpretations exist, leading to an ambiguity error.

Q-4

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
 public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

 INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 INR(USD u);

 operator float(){
 return (rs + ps/100.0);
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }

};

class USD{
 int dlr;
 int cnt;
 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};

INR::INR(USD u){
 *this = INR(float(u)*exchange);
}

int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2 = io1;
 cout << "uo2 : ";
 uo2.print();

 float x = io1 * 2 ;
 float y = 2 * io2 ;
 float z = uo1 * 2 ;
 float p = 2 * uo2 ;

 cout << fixed << setprecision(2) << "X:"
 << x << " Y:" << y << endl << "z:" <<
 z << " P:" << p << endl;
 return 0;
}
```

output ?

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
 public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

 INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 INR(USD u);

 operator float(){
 return (rs + ps/100.0);
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }

};
```

```
class USD{
 int dlr;
 int cnt;
 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```
INR::INR(USD u){
 *this = INR(float(u)*exchange);
}

int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2 = io1;
 cout << "uo2 : ";
 uo2.print();

 float x = io1 * 2 ;
 float y = 2 * io2 ;
 float z = uo1 * 2 ;
 float p = 2 * uo2 ;

 cout << fixed << setprecision(2) << "X:"
 << x << " Y:" << y << endl << "z:" <<
 z << " P:" << p << endl;
 return 0;
}
```

OUTPUT :

```
 io1 : INR = 358.50
uo1 : USD = 38.54
 io2 : INR = 3293.63
 io3 : INR = 3293.63
uo2 : USD = 4.19
X : 717.00   Y : 6587.26
Z : 77.08     P : 8.38
```

Maulik N. Patoliya

Q-5

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
 public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

 INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 INR(USD u);

 operator float(){
 return (rs + ps/100.0);
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }
};
```

```
class USD{
 int dlr;
 int cnt;
 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```
INR::INR(USD u){
 *this = INR(float(u)*exchange);
}

int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2 = io1;
 cout << "uo2 : ";
 uo2.print();

 INR io4 = io1 * 2 ;
 USD uo3 = uo1 * 2 ;

 cout << "io4 : ";
 io4.print();
 cout << "uo3 : ";
 uo3.print();
 return 0;

}
```

output ?

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
 public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

 INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 INR(USD u);

 operator float(){
 return (rs + ps/100.0);
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }
};
```

```
class USD{
 int dlr;
 int cnt;
 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```
INR::INR(USD u){
 *this = INR(float(u)*exchange);
}

int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2 = io1;
 cout << "uo2 : ";
 uo2.print();

 INR io4 = io1 * 2 ;
 USD uo3 = uo1 * 2 ;

 cout << "io4 : ";
 io4.print();
 cout << "uo3 : ";
 uo3.print();
 return 0;

}
```

OUTPUT :

```
 io1 : INR = 358.50
uo1 : USD = 38.54
 io2 : INR = 3293.63
 io3 : INR = 3293.63
uo2 : USD = 4.19
 io4 : INR = 717.0
uo3 : USD = 77.8
```

Maulik N. Patoliya

Q-6

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
 public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

 INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 INR(USD u);

 operator float(){
 return (rs + ps/100.0);
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }
};
```

```
class USD{
 int dlr;
 int cnt;
 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```
INR::INR(USD u){
 *this = INR(float(u)*exchange);
}

int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2 = io1;
 cout << "uo2 : ";
 uo2.print();

 INR io4 = io1 * 2 * uo1;

 USD uo3 = uo1 * 2 * io2;

 cout << "io4 : ";
 io4.print();
 cout << "uo3 : ";
 uo3.print();
 return 0;

}
```

OUTPUT :

```
 io1 : INR = 358.50
uo1 : USD = 38.54
 io2 : INR = 3293.63
 io3 : INR = 3293.63
uo2 : USD = 4.19
 io4 : INR = 27633.18
uo3 : USD = 253873.0
```

➩ Code doesn’t throw any error but we don’t get correct output of io4 and uo3.

Here , uo1 was not converted into INR

Here , io2 was not converted into USD

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
 public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

 INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 INR(USD u);

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator*(int a){
 return INR(float(*this) * a);
 }

 void print(){
 cout << fixed << "INR = " << rs << "."
 << ps << endl;
 }
};
```

```
class USD{
 int dlr;
 int cnt;
 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 USD operator*(int a){
 return USD(float(*this) * a);
 }

 void print(){
 cout << fixed << "USD = " << dlr << "."
 << cnt << endl;
 }

};
```

```
INR::INR(USD u){
 *this = INR(float(u)*exchange);
}

INR operator*( INR i , USD u){
 return INR( float(i) * (float(u)*exchange));
}

USD operator*(USD u , INR i){
 return USD( float(u) * (float(i)/exchange) );
}

int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();

 USD uo2 = io1;
 cout << "uo2 : ";
 uo2.print();

 INR io4 = io1 * 2 * uo1;

 USD uo3 = uo1 * 2 * io2;
```

```
 cout << "io4 : ";
 io4.print();
 cout << "uo3 : ";
 uo3.print();
 return 0;
}
```

OUTPUT :

```
 io1 : INR = 358.50
uo1 : USD = 38.54
 io2 : INR = 3293.63
 io3 : INR = 3293.63
uo2 : USD = 4.19
 io4 : INR = 2361531.50
uo3 : USD = 2970.66
```

Maulik N. Patoliya

Q-7

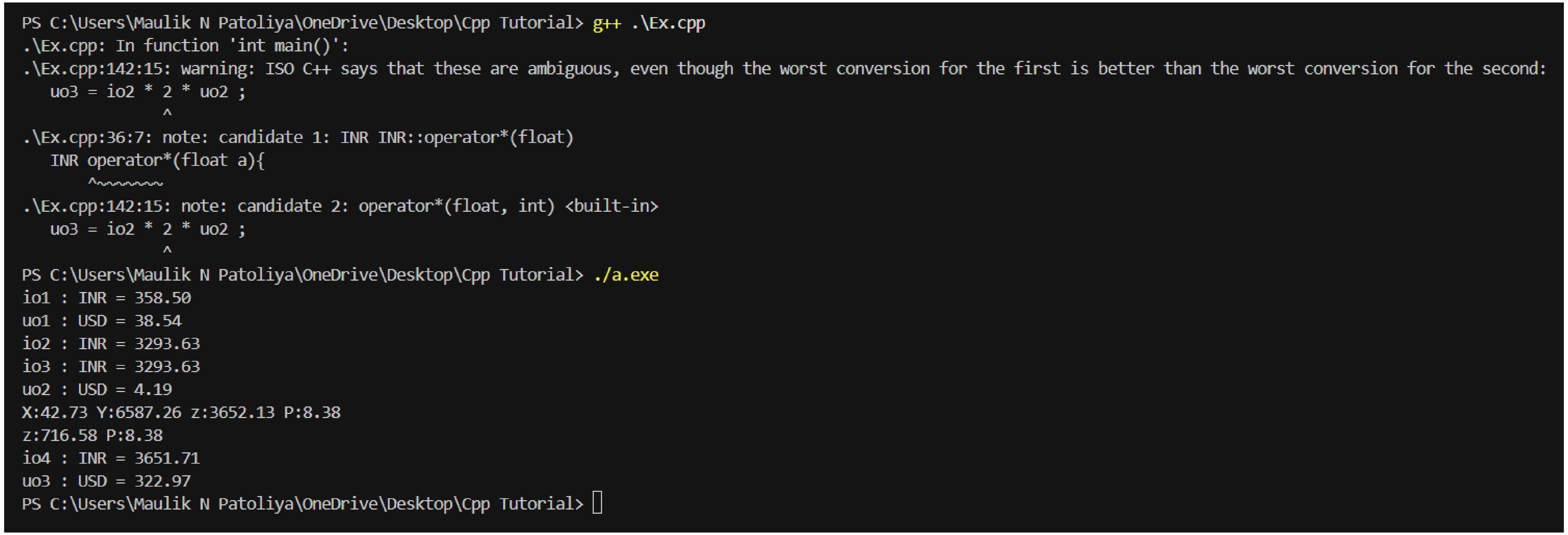
```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
 public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

 INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 INR(USD u);

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator+( INR io2){
 return INR( float(*this) + float(io2) );
 }

 INR operator*(float a){
 return INR( float(*this) * a );
 }
```

```
void print(){
 cout << fixed << "INR = " << rs
<< "." << ps << endl;
 }

};

class USD{
 int dlr;
 int cnt;

 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 USD operator+( USD uo2){
 return USD( float(*this) + float(uo2) );
 }

 USD operator*(float a){
 return USD( float(*this) * a );
 }
```

```
void print(){
 cout << fixed << "USD = " << dlr
 << "."<< cnt << endl;
 }

};

INR::INR(USD u){
 *this = INR(float(u)*exchange);
}

INR operator+( INR i , USD u){
 return INR( float(i) + (float(u)*exchange));
}

USD operator+( USD u , INR i ){
 return USD( float(u) + (float(i)/exchange));
}


int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();
```

```
USD uo2(io1);
 cout << "uo2 : ";
 uo2.print();

 float x = uo1 + uo2 ;
 float y = 2 * io2;
 float z = io1 + io2 ;
 float p = 2 * uo2 ;

 cout << fixed << setprecision(2) <<
 "X:" << x << " Y:" << y << " z:" <<
 z << " P:" << p << endl;

 z = io1 + uo2 ;
 p = uo2 + io1 ;

 cout << fixed << setprecision(2) <<
 "z:" << z << " P:" << p << endl;

 INR io4 ;
 io4 = uo2 + io3 ;

 USD uo3;
 uo3 = io2 * 2 * uo2 ;

 cout << "io4 : ";
 io4.print();
 cout << "uo3 : ";
 uo3.print();

 return 0;
}
```

This code would be give correct output but generate some warnings :



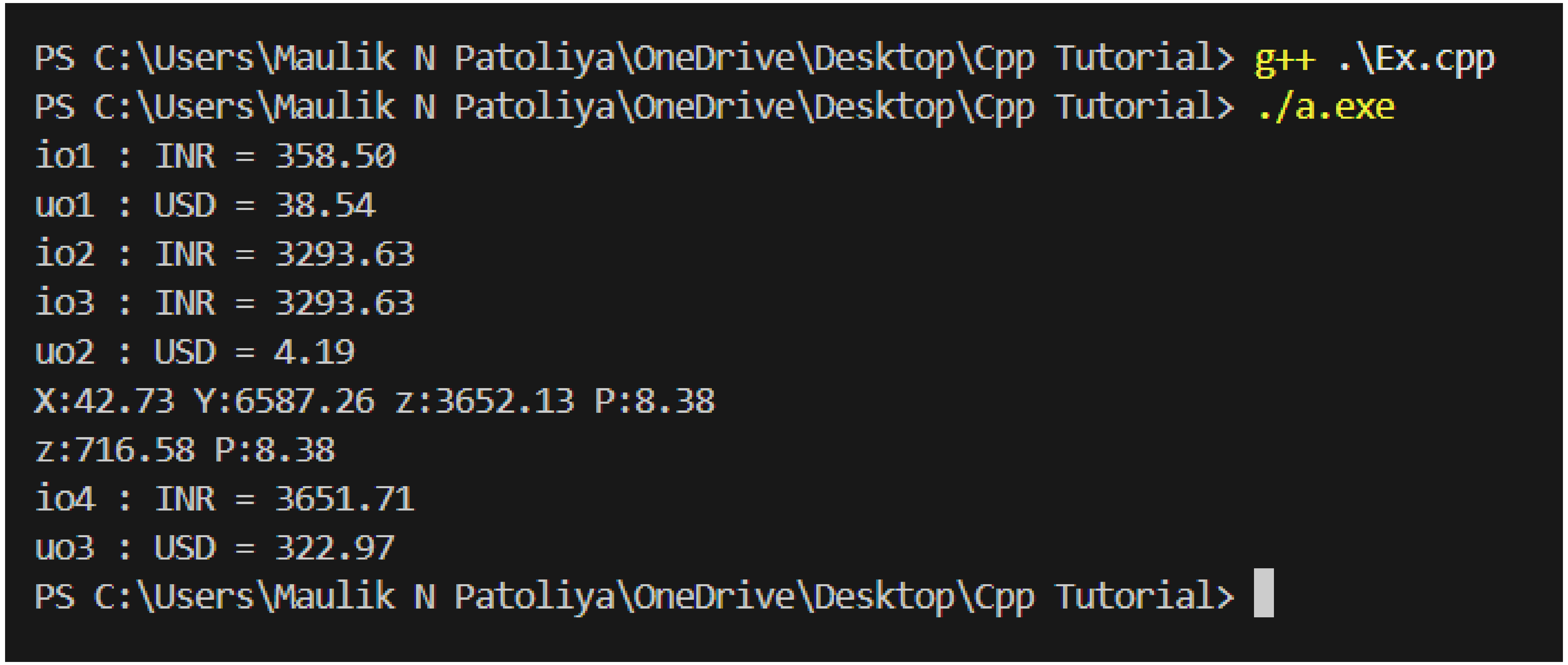
```
PS C:\Users\Maulik N Patoliya\OneDrive\Desktop\Cpp Tutorial> g++ .\Ex.cpp
.\Ex.cpp: In function 'int main()':
.\Ex.cpp:142:15: warning: ISO C++ says that these are ambiguous, even though the worst conversion for the first is better than the worst conversion for the second:
   uo3 = io2 * 2 * uo2 ;
               ^
.\Ex.cpp:36:7: note: candidate 1: INR INR::operator*(float)
   INR operator*(float a){
       ^~~~~~~~
.\Ex.cpp:142:15: note: candidate 2: operator*(float, int) <built-in>
   uo3 = io2 * 2 * uo2 ;
               ^
PS C:\Users\Maulik N Patoliya\OneDrive\Desktop\Cpp Tutorial> ./a.exe
io1 : INR = 358.50
uo1 : USD = 38.54
io2 : INR = 3293.63
io3 : INR = 3293.63
uo2 : USD = 4.19
X:42.73 Y:6587.26 z:3652.13 P:8.38
z:716.58 P:8.38
io4 : INR = 3651.71
uo3 : USD = 322.97
PS C:\Users\Maulik N Patoliya\OneDrive\Desktop\Cpp Tutorial> 
```

Solution : →

Maulik N. Patoliya

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define exchange 85.46
using namespace std;

class USD;

class INR{
 int rs;
 int ps;
 public:

 INR():rs(0),ps(0){}

 INR(float r){
 rs = r;
 ps = round((r-rs)*100);
 }

 INR(int r , int p){
 rs = r + p/100 ;
 ps = p%100;
 }

 INR(USD u);

 operator float(){
 return (rs + ps/100.0);
 }

 INR operator+( INR io2){
 return INR( float(*this) + float(io2) );
 }

 /* INR operator*(float a){
 return INR( float(*this) * a );
 } */

void print(){
 cout << fixed << "INR = " << rs
<< "." << ps << endl;
 }

};

class USD{
 int dlr;
 int cnt;

 public:

 USD():dlr(0),cnt(0){}

 USD(float d){
 dlr = d;
 cnt = round((d-dlr)*100);
 }

 USD(int d , int c){
 dlr = d + c/100;
 cnt = c%100;
 }

 USD(INR i){
 *this = USD(i/exchange);
 }

 operator float(){
 return ( dlr + cnt/100.0);
 }

 USD operator+( USD uo2){
 return USD( float(*this) + float(uo2) );
 }

 /* USD operator*(float a){
 return USD( float(*this) * a );
 } */

void print(){
 cout << fixed << "USD = " << dlr
 << "."<< cnt << endl;
 }

};

INR::INR(USD u){
 *this = INR(float(u)*exchange);
}

INR operator+( INR i , USD u){
 return INR( float(i) + (float(u)*exchange));
}

USD operator+( USD u , INR i ){
 return USD( float(u) + (float(i)/exchange));
}


int main(){

 INR io1(358.5);
 USD uo1(38.538);

 cout << "io1 : ";
 io1.print();
 cout << "uo1 : ";
 uo1.print();

 INR io2(uo1);
 INR io3;
 io3 = uo1;

 cout << "io2 : ";
 io2.print();
 cout << "io3 : ";
 io3.print();
USD uo2(io1);
 cout << "uo2 : ";
 uo2.print();

 float x = uo1 + uo2 ;
 float y = 2 * io2;
 float z = io1 + io2 ;
 float p = 2 * uo2 ;

 cout << fixed << setprecision(2) <<
 "X:" << x << " Y:" << y << " z:" <<
 z << " P:" << p << endl;

 z = io1 + uo2 ;
 p = uo2 + io1 ;

 cout << fixed << setprecision(2) <<
 "z:" << z << " P:" << p << endl;

 INR io4 ;
 io4 = uo2 + io3 ;

 USD uo3;
 uo3 = USD(io2) * 2 * uo2 ;

 cout << "io4 : ";
 io4.print();
 cout << "uo3 : ";
 uo3.print();

 return 0;
}
```

output :



```
PS C:\Users\Maulik N Patoliya\OneDrive\Desktop\Cpp Tutorial> g++ .\Ex.cpp
PS C:\Users\Maulik N Patoliya\OneDrive\Desktop\Cpp Tutorial> ./a.exe
io1 : INR = 358.50
uo1 : USD = 38.54
io2 : INR = 3293.63
io3 : INR = 3293.63
uo2 : USD = 4.19
X:42.73 Y:6587.26 z:3652.13 P:8.38
z:716.58 P:8.38
io4 : INR = 3651.71
uo3 : USD = 322.97
PS C:\Users\Maulik N Patoliya\OneDrive\Desktop\Cpp Tutorial>
```

Q-8

At what time which function should be called..?

```
#include <iostream>
#include <iomanip>
#include <cmath>
/* 1 feet = 0.3048 meterr */
#define exchange 0.3048
using namespace std;

class DB;

class DM{
 int meter;
 int cm;
 public:

 DM():meter(0),cm(0){
  cout << "DM : 0 argumented" << endl;
 }

 DM(float d){
  cout << "DM : 1 argumented" << endl;
  meter = d;
  cm = (d-meter)*100;
 }
 DM(int meter , int cm){
  cout << "Dm : 2 argumented" << endl;
  this->meter = meter + cm/100;
  this->cm = cm % 100;
 }
 DM(DB obj);

 operator float(){
  cout << "DM : float() overload" << endl;
  return (meter + cm/100.0);
 }

 DM operator+(DM ob2){
  cout << "DM member +" << endl;
  return DM( float(*this) + float(ob2) );
 }

 void print(){
  cout << fixed << "Meter.cm = " << meter <<
  "." << cm << endl;
 }
};

class DB{
 int feet;
 int inch;
 public:

 DB():feet(0),inch(0){
  cout << "DB : 0 argumented" << endl;
 }

 DB(float d){
  cout << "DB : 1 argumented" << endl;
  feet = d;
  inch = (d-feet)*12;
 }

 DB(int feet , int inch){
  cout << "DB : 2 argumented" << endl;
  feet = feet + inch/12;
  inch = inch % 12;
 }

 DB(DM obj){
  cout << "DB : convert DM->DB" << endl;
  *this = DB( float(obj) / exchange );
 }

 operator float(){
  cout << "DB : float() overload" << endl;
  return (feet + inch/12.0);
 }

 DB operator+(DB ob2){
  cout << "DM member +" << endl;
  return DB( float(*this) + float(ob2) );
 }

 void print(){
  cout << fixed << "Feet.inch = " << feet <<
  "." << inch << endl;
 }
};

DM::DM(DB obj){
 cout << "DM : convert DB->DM" << endl;
 *this = DM ( float(obj) * exchange );
}

DM operator+(DM obj1 , DB obj2){
 cout << "DM non-member +" << endl;
 return DM( float(obj1) + (float(obj2)*exchange) );
}

DB operator+(DB obj1 , DM obj2){
 cout << "DB non-member +" << endl;
 return DB( float(obj1) + (float(obj2)/exchange) );
}


int main(){
 DM m1(25 , 56);
 DB b1(12.75);

 cout << "m1 : " ;
 m1.print();
 cout << "b1 : " ;
 b1.print();

 cout << endl;
 DM m2(b1);
 DM m3 = b1;
 cout << "m2 : ";
 m2.print();
 cout << "m3 : ";
 m3.print();

cout << endl;
 DB b2(m1);
 cout << "b2 : ";
 b2.print();

 cout << endl;
 float x = m1 + m2 ;
 float y = 2 * m2;
 float z = b1 + b2 ;
 float p = 2 * b2 ;

 cout << fixed << setprecision(2) <<
 "X:" << x << " Y:" << y << " z:" <<
 z << " P:" << p << endl << endl;

 z = m1 + b2 ;
 p = b2 + m1 ;

 cout << fixed << setprecision(2) <<
 "z:" << z << " P:" << p << endl << endl;

 DM m4 ;
 m4 = b2 + m3 ;
 DB b3;
 b3 = DB(m2) * 2 * b2 ;

 cout << "m4 : ";
 m4.print();
 cout << "b3 : ";
 b3.print();

 return 0;
}
```

output ?

Maulik N. Patoliya

```
#include <iostream>
#include <iomanip>
#include <cmath>
/* 1 feet = 0.3048 meterr */
#define exchange 0.3048
using namespace std;

class DB;

class DM{
 int meter;
 int cm;
 public:

 DM():meter(0),cm(0){
  cout << "DM : 0 argumented" << endl;
 }

 DM(float d){
  cout << "DM : 1 argumented" << endl;
  meter = d;
  cm = (d-meter)*100;
 }
 DM(int meter , int cm){
  cout << "Dm : 2 argumented" << endl;
  this->meter = meter + cm/100;
  this->cm = cm % 100;
 }
 DM(DB obj);

 operator float(){
  cout << "DM : float() overload" << endl;
  return (meter + cm/100.0);
 }

 DM operator+(DM ob2){
  cout << "DM member +" << endl;
  return DM( float(*this) + float(ob2) );
 }
 void print(){
  cout << fixed << "Meter.cm = " << meter <<
  "." << cm << endl;
 }
};

class DB{
 int feet;
 int inch;
 public:

 DB():feet(0),inch(0){
  cout << "DB : 0 argumented" << endl;
 }

 DB(float d){
  cout << "DB : 1 argumented" << endl;
  feet = d;
  inch = (d-feet)*12;
 }

 DB(int feet , int inch){
  cout << "DB : 2 argumented" << endl;
  feet = feet + inch/12;
  inch = inch % 12;
 }

 DB(DM obj){
  cout << "DB : convert DM->DB" << endl;
  *this = DB( float(obj) / exchange );
 }

 operator float(){
  cout << "DB : float() overload" << endl;
  return (feet + inch/12.0);
 }

 DB operator+(DB ob2){
  cout << "DM member +" << endl;
  return DB( float(*this) + float(ob2) );
 }
 void print(){
  cout << fixed << "Feet.inch = " << feet <<
  "." << inch << endl;
 }
};

DM::DM(DB obj){
 cout << "DM : convert DB->DM" << endl;
 *this = DM ( float(obj) * exchange );
}

DM operator+(DM obj1 , DB obj2){
 cout << "DM non-member +" << endl;
 return DM( float(obj1) + (float(obj2)*exchange) );
}

DB operator+(DB obj1 , DM obj2){
 cout << "DB non-member +" << endl;
 return DB( float(obj1) + (float(obj2)/exchange) );
}

int main(){
 DM m1(25,56);
 DB b1(12.75);

 cout << "m1 : " ;
 m1.print();
 cout << "b1 : " ;
 b1.print();

 coul << endl << "m2" << endl;
 DM m2(b1);
 cout << "m3" << endl;
 DM m3 = b1;
 cout << "m2 : ";
 m2.print();
 cout << "m3 : ";
 m3.print();
cout << endl << "b2" << endl;
 DB b2(m1);
 cout << "b2 : ";
 b2.print();

 cout << endl << "x" << endl;
 float x = m1 + m2 ;
 cout << "y" << endl;
 float y = 2 * m2;
 cout << "z" << endl;
 float z = b1 + b2 ;
 cout << "p" << endl;
 float p = 2 * b2 ;

 cout << fixed << setprecision(2) <<
 "X:" << x << " Y:" << y << " z:" <<
 z << " P:" << p << endl << endl;

 cout << "z" << endl;
 z = m1 + b2 ;
 cout << "p" << endl;
 p = b2 + m1 ;

 cout << fixed << setprecision(2) <<
 "z:" << z << " P:" << p << endl << endl;

 cout << "m4" << endl;
 DM m4 ;
 m4 = b2 + m3 ;
 cout << "b3" << endl;
 DB b3;
 b3 = DB(m2) * 2 * b2 ;

 cout << "m4 : ";
 m4.print();
 cout << "b3 : ";
 b3.print();

 return 0;
}
```

OUTPUT :

```
DM : 2 argumented
DB  : 1 argumented
m1 : Meter.cm = 25.56
b1  : Feet.inch = 12.9

m2
DM : convert DB->DM
DB  : float() overload
DM : 1 argumented
m3
DM : convert DB->DM
DB  : float() overload
DM : 1 argumented
m2 : Meter.cm = 3.88
m3 : Meter.cm = 3.88

b2
DB  : convert DM->DB
DM : float() overload
DB  : 1 argumented
b2   : Feet.inch = 83.10

x
DM member +
DM : float() overload
DM : float() overload
DM : 1 argumented
DM : float() overload
```

```
y
DM : float() overload
z
DM member +
DB : float() overload
DB : float() overload
DB : 1 argumented
DB : float() overload
p
DB : float() overload
X:29.43 Y:7.76 z:96.58 P:167.67

z
DM non-member +
DM : float() overload
DB  : float() overload
DM : 1 argumented
DM : float() overload
p
DB non-member +
DB  : float() overload
DM : float() overload
DB  : 1 argumented
DB  : float() overload
z:51.11   P:167.67
```

```
m4
DM : 0 argumented
DB non-member +
DB  : float() overload
DM : float() overload
DB  : 1 argumented
DM : convert DB->DM
DB  : float() overload
DM : 1 argumented
b3
DB  : 0 argumented
DB  : convert DM->DB
DM : float() overload
DB  : 1 argumented
DB  : float() overload
DB  : float() overload
DB  : 1 argumented
m4  : Meter.cm = 29.41
b3   : Feet.inch = 2123.9
```

Dry Code is....

```
#include <iostream>
#include <cmath>
#include <iomanip>
/* 1 feet = 0.3048 meter */
#define exchange 0.3048
using namespace std;

class DB;

class DM{
 int meter;
 int cm;
 public:

 DM():meter(0),cm(0){}

 DM(float d){
  meter = d;
  cm = (d-meter)*100;
 }
 DM(int meter , int cm){
  this->meter = meter + cm/100;
  this->cm = cm % 100;
 }
 DM(DB obj);

 operator float(){
  return (meter + cm/100.0);
 }

 DM operator+(DM ob2){
  return DM( float(*this) + float(ob2) );
 }

 void print(){
  cout << fixed << "Meter.cm = " << meter <<
  "." << cm << endl;
 }
};
```

```
class DB{
 int feet;
 int inch;
 public:

 DB():feet(0),inch(0){}

 DB(float d){
  feet = d;
  inch = (d-feet)*12;
 }

 DB(int feet , int inch){
  feet = feet + inch/12;
  inch = inch % 12;
 }

 DB(DM obj){
  *this = DB( float(obj) / exchange );
 }

 operator float(){
  return (feet + inch/12.0);
 }

 DB operator+(DB ob2){
  return DB( float(*this) + float(ob2) );
 }

 void print(){
  cout << fixed << "Feet.inch = " << feet <<
  "." << inch << endl;
 }
};

DM::DM(DB obj){
 *this = DM ( float(obj) * exchange );
}
```

```
DM operator+(DM obj1 , DB obj2){
 return DM( float(obj1) + (float(obj2)*exchange) );
}

DB operator+(DB obj1 , DM obj2){
 return DB( float(obj1) + (float(obj2)/exchange) );
}


int main(){
 DM m1(25,56);
 DB b1(12.75);

 cout << "m1 : " ;
 m1.print();
 cout << "b1 : " ;
 b1.print();

 cout << endl;
 DM m2(b1);
 DM m3 = b1;
 cout << "m2 : ";
 m2.print();
 cout << "m3 : ";
 m3.print();

 cout << endl;
 DB b2(m1);
 cout << "b2 : ";
 b2.print();

 cout << endl;
 float x = m1 + m2 ;
 float y = 2 * m2;
 float z = b1 + b2 ;
 float p = 2 * b2 ;
```

```
cout << fixed << setprecision(2) <<
 "X:" << x << " Y:" << y << "\nz:" <<
 z << " P:" << p << endl << endl;

 z = m1 + b2 ;
 p = b2 + m1 ;

 cout << fixed << setprecision(2) <<
 "z:" << z << " P:" << p << endl << endl;

 DM m4 ;
 m4 = b2 + m3 ;
 DB b3;
 b3 = DB(m2) * 2 * b2 ;

 cout << "m4 : ";
 m4.print();
 cout << "b3 : ";
 b3.print();

 return 0;
}
```

OUTPUT :

```
 m1 : Meter.cm = 25.56
b1 : Feet.inch = 12.9

m2 : Meter.cm = 3.88
m3 : Meter.cm = 3.88

b2 : Feet.inch = 83.10

X:29.43  Y:7.76
z:96.58   P:167.67

z:51.11 P:167.67

m4 : Meter.cm = 29.41
b3 : Feet.inch = 2123.9
```

Maulik N. Patoliya

Q-9

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define PI 3.14
using namespace std;

/*
  cartesian Coordinates -- ctn (x,y)
  Polar Coordinates -- plr (r,θ)
  r = (x^2 + y^2)^(1/2);
  θ = tan^-1(y/x); θ --> should be in radian
  similar θ = atan2(y,x) --> [-π, π]
*/

class plr;

class ctn{
  double x;
  double y;
  public:

  ctn():x(0),y(0){}

  ctn(double x, double y):x(x),y(y){}

  ctn(plr p);

  double get_x(){
  return x;
  }

  double get_y(){
  return y;
  }

  ctn operator+(ctn o){
  return ctn( (*this).x + o.x , (*this).y + o.y );
  }

ctn operator-(ctn o){
  return ctn( (*this).x - o.x , (*this).y - o.y );
  }

  ctn operator*(ctn a){
   return ctn( x*a.x , y*a.y);
  }

  void print(){
  cout << fixed << setprecision(2) << "(x,y) : "
  << x << "," << y << endl;
  }

};

class plr{
  double r;
  double t; // t = θ (theta);
  public:

  plr():r(0),t(0){}

  plr(double r, double t):r(r),t(t){}

  plr(ctn ob){
    double x = ob.get_x();
    double y = ob.get_y();

   *this = plr(sqrt(x*x + y*y),atan2(y,x));
  }

  double get_r(){
   return r;
  }

  double get_t(){
   return t;
  }

plr operator+(plr ob){
   double x = r*cos(t) + ob.r*cos(ob.t);
   double y = r*sin(t) + ob.r*sin(ob.t);
   return plr(sqrt(x*x + y*y),atan2(y,x));
  }

  plr operator-(plr ob){
   double x = r*cos(t) - ob.r*cos(ob.t);
   double y = r*sin(t) - ob.r*sin(ob.t);
   return plr(sqrt(x*x + y*y),atan2(y,x));
  }

  plr operator*(plr p){
   double n_r = r * p.r;
   double n_t = t + p.t;

   // Angale should between -- [-π, π]
   while (n_t > PI) n_t -= 2 * PI;
   while (n_t <= -PI) n_t += 2 * PI;

   return plr(n_r, n_t);
  }

  void print(){
   cout << fixed << setprecision(2) <<
   "(r,t) : " << r << "," << t << endl;
  }

};
ctn::ctn(plr ob){
   *this = ctn(ob.get_r()*cos(ob.get_t()) ,
         ob.get_r()*sin(ob.get_t()));
}
```

Maulik N. Patoliya

```
ctn operator+(ctn a , plr b){
 return ctn( a.get_x() + b.get_r()*cos(b.get_t()) ,
 a.get_y() + b.get_r()*sin(b.get_t()) );
}

plr operator+(plr a , ctn b){
  double x = a.get_r() * cos(a.get_t()) + b.get_x();
  double y = a.get_r() * sin(a.get_t()) + b.get_y();
  return plr(sqrt(x*x + y*y),atan2(y,x));
}

ctn operator*(ctn a , plr b){
  return ctn( a.get_x() * b.get_r()*cos(b.get_t()) ,
        a.get_y() * b.get_r()*sin(b.get_t()) );
}

plr operator*(plr a , ctn b){
  double x = a.get_r() * cos(a.get_t()) * b.get_x();
  double y = a.get_r() * sin(a.get_t()) * b.get_y();
  return plr(sqrt(x*x + y*y),atan2(y,x));
}

int main(){
  ctn c1(2,5);
  plr p1(7.07,0.79);

  cout << "c1 = " ;
  c1.print();
  cout << "p1 = " ;
  p1.print();
```

```
 cout << endl;
 ctn c2(p1);
 ctn c3 = p1;
 cout << "c2 = ";
 c2.print();
 cout << "c3 = ";
 c3.print();

cout << endl;
  ctn c4 = p2 + c3 ;
  plr p3 = p2 - p1 ;
  cout << "c4 = ";
  c4.print();
  cout << "p3 = ";
  p3.print();

  cout << endl;
  ctn c5 = c3 * c4;
  plr p4 = p2 * p3;
  cout << "c5 = ";
  c5.print();
  cout << "p4 = ";
  p4.print();

  cout << endl;
  ctn c6 = c3 * p4;
  plr p5 = p2 * c4;
  cout << "c6 = ";
  c6.print();
  cout << "p5 = ";
  p5.print();

  return 0;
}
```

output ?

```
#include <iostream>
#include <cmath>
#include <iomanip>
#define PI 3.14
using namespace std;

/*
  cartesian Coordinates -- ctn (x,y)
  Polar Coordinates -- plr (r,θ)
  r = (x^2 + y^2)^(1/2);
  θ = tan^-1(y/x); θ --> should be in radian
  similar θ = atan2(y,x) --> [-π, π]
*/

class plr;

class ctn{
  double x;
  double y;
  public:

  ctn():x(0),y(0){}

  ctn(double x, double y):x(x),y(y){}

  ctn(plr p);

  double get_x(){
  return x;
  }

  double get_y(){
  return y;
  }

  ctn operator+(ctn o){
  return ctn( (*this).x + o.x , (*this).y + o.y );
  }
```

```
ctn operator-(ctn o){
  return ctn( (*this).x - o.x , (*this).y - o.y );
  }

  ctn operator*(ctn a){
   return ctn( x*a.x , y*a.y);
  }

  void print(){
  cout << fixed << setprecision(2) << "(x,y) : "
  << x << "," << y << endl;
  }

};

class plr{
  double r;
  double t; // t = θ (theta);
  public:

  plr():r(0),t(0){}

  plr(double r, double t):r(r),t(t){}

  plr(ctn ob){
    double x = ob.get_x();
    double y = ob.get_y();

   *this = plr(sqrt(x*x + y*y),atan2(y,x));
  }

  double get_r(){
   return r;
  }

  double get_t(){
   return t;
  }
```

```
plr operator+(plr ob){
   double x = r*cos(t) + ob.r*cos(ob.t);
   double y = r*sin(t) + ob.r*sin(ob.t);
   return plr(sqrt(x*x + y*y),atan2(y,x));
  }

  plr operator-(plr ob){
   double x = r*cos(t) - ob.r*cos(ob.t);
   double y = r*sin(t) - ob.r*sin(ob.t);
   return plr(sqrt(x*x + y*y),atan2(y,x));
  }

  plr operator*(plr p){
   double n_r = r * p.r;
   double n_t = t + p.t;

   // Angale should between -- [-π, π]
   while (n_t > PI) n_t -= 2 * PI;
   while (n_t <= -PI) n_t += 2 * PI;

   return plr(n_r, n_t);
  }

  void print(){
   cout << fixed << setprecision(2) <<
   "(r,t) : " << r << "," << t << endl;
  }

};
ctn::ctn(plr ob){
   *this = ctn(ob.get_r()*cos(ob.get_t()) ,
         ob.get_r()*sin(ob.get_t()));
}
```

```
ctn operator+(ctn a , plr b){
 return ctn( a.get_x() + b.get_r()*cos(b.get_t()) ,
 a.get_y() + b.get_r()*sin(b.get_t()) );
}

plr operator+(plr a , ctn b){
  double x = a.get_r() * cos(a.get_t()) + b.get_x();
  double y = a.get_r() * sin(a.get_t()) + b.get_y();
  return plr(sqrt(x*x + y*y),atan2(y,x));
}

ctn operator*(ctn a , plr b){
  return ctn( a.get_x() * b.get_r()*cos(b.get_t()) ,
        a.get_y() * b.get_r()*sin(b.get_t()) );
}

plr operator*(plr a , ctn b){
  double x = a.get_r() * cos(a.get_t()) * b.get_x();
  double y = a.get_r() * sin(a.get_t()) * b.get_y();
  return plr(sqrt(x*x + y*y),atan2(y,x));
}

int main(){
  ctn c1(2,5);
  plr p1(7.07,0.79);

  cout << "c1 = " ;
  c1.print();
  cout << "p1 = " ;
  p1.print();
```

```
 cout << endl;
 ctn c2(p1);
 ctn c3 = p1;
 cout << "c2 = ";
 c2.print();
 cout << "c3 = ";
 c3.print();

cout << endl;
  ctn c4 = p2 + c3 ;
  plr p3 = p2 - p1 ;
  cout << "c4 = ";
  c4.print();
  cout << "p3 = ";
  p3.print();

  cout << endl;
  ctn c5 = c3 * c4;
  plr p4 = p2 * p3;
  cout << "c5 = ";
  c5.print();
  cout << "p4 = ";
  p4.print();

  cout << endl;
  ctn c6 = c3 * p4;
  plr p5 = p2 * c4;
  cout << "c6 = ";
  c6.print();
  cout << "p5 = ";
  p5.print();

  return 0;
}
```

OUTPUT :

```
c1 = (x,y) : 2.00,5.00
p1 = (r,t) : 7.07,0.79

c2 = (x,y) : 4.98,5.02
c3 = (x,y) : 4.98,5.02

p2 = (r,t) : 5.39,1.19

c4 = (x,y) : 6.98,10.02
p3 = (r,t) : 2.98,-3.13

c5 = (x,y) : 34.71,50.33
p4 = (r,t) : 16.03,-1.94

c6 = (x,y) : -29.07,-74.96
p5 = (r,t) : 52.02,1.30
```