**Sunny kumar**

**Ambiguity in function overloading**

**[1]**

```
#include<iostream>
using namespace std;
void test(float a, float b)
{
      cout<<"x is"<<a<<endl<<"y is"<<b<<endl;
}
void test(int a, int b)
{

   cout<<"x is"<<a<<endl<<"y is"<<b<<endl;

}

int main()
{
   float x=5.5,y=6.9;
    test(5.5,6.9);// error of ambiguity bcz compiler doesn't know which function is called
    test(5.5,6.9);
    return 0;

}
```

**[2]**

```
#include<iostream>
using namespace std;
void test(int a,int b)
{
      cout<<"x is"<<a<<endl<<"y is"<<b<<endl;
}
void test(int& a, int& b)
{

   cout<<"x is"<<a<<endl<<"y is"<<b<<endl;

}

int main()
{
```

```
    int x=5,y=6;
     test(x,y);
     test(x,y);
     return 0;

}
```

**[3] Ambiguity with default arguments**

```
#include<iostream>
using namespace std;
void test(int a,int b=10)          //default arguments
{
      cout<<"a is"<<a<<endl<<"b is"<<b;
}
void test(int a=6.5)
{

    cout<<"x is"<<a<<endl;

}

int main()
{
    int x=5,y=6;
     test(5.5);
     test(5.5);

     return 0;

}
```

**[4]**

```
#include <iostream>
#include <cmath>
using namespace std;
class Shape {
public:
    double calculateArea(double length, double width) {
        return length * width;
    }
    double calculateArea(double radius) {
        return M_PI * radius * radius;
    }
```

```
    double calculateArea(double base, double height, bool isTriangle) {
        if (isTriangle){
            return 0.5 * base * height;    //   area of triangle=1/2*base*height
        }else{
            return 0.0;
        }
    }
};

int main() {
    Shape findshape;
    double rectangleArea = findshape.calculateArea(6, 3.0);
    cout << "Area of Rectangle is: " << rectangleArea << endl;
    double circleArea = findshape.calculateArea(4.0);
    cout << "Area of Circle is: " << circleArea << endl;
    double triangleArea = findshape.calculateArea(4.0, 3.0, true);
    cout << "Area of Triangle is: " << triangleArea << endl;
    return 0;
}
```

**[5]**

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

class Distance {
private:
    int feet;
    int inches;

public:

    Distance(int f = 0, int in = 0) : feet(f), inches(in) {
        normalize();
    }

    void normalize() {
        if (inches >= 12) {
            feet += inches / 12;
            inches = inches % 12;
```

```
    } else if (inches < 0) {
        int borrow = (-inches + 11) / 12;
        feet -= borrow;
        inches += borrow * 12;
    }
}

void display() const {
    cout << feet << " feet " << inches << " inches" << endl;
}

Distance operator+(const Distance& d) const {
    int totalFeet = feet + d.feet;
    int totalInches = inches + d.inches;
    return Distance(totalFeet, totalInches);
}

Distance operator-(const Distance& d) const {
    int totalFeet = feet - d.feet;
    int totalInches = inches - d.inches;
    return Distance(totalFeet, totalInches);
}

bool operator==(const Distance& d) const {
    return feet == d.feet && inches == d.inches;
}

bool operator!=(const Distance& d) const {
    return !(*this == d);
}

bool operator<(const Distance& d) const {
    int thisTotal = feet * 12 + inches;
    int dTotal = d.feet * 12 + d.inches;
    return thisTotal < dTotal;
}

bool operator>(const Distance& d) const {
    return d < *this;
}

bool operator<=(const Distance& d) const {
    return !(d < *this);
}
```

```
    bool operator>=(const Distance& d) const {
        return !(*this < d);
    }
};

int main() {
    Distance d1(5, 8);
    Distance d2(3, 10);

    Distance sum = d1 + d2;
    Distance diff = d1 - d2;

    cout << "Sum: ";
    sum.display();

    cout << "Difference: ";
    diff.display();

    if (d1 > d2) {
        cout << "d1 is greater than d2" << endl;
    } else if(d1 == d2) {
        cout << "d1 is not less than d2" << endl;
    }else{
        cout<<"d2 is greater than d1"<<endl;
    }



    return 0;
}
```

**[7]**

```
#include <iostream>
using namespace std;
class animal{
    public:
    void eat()
    {
        cout<<"eating"<<endl;
    }
```

```
};
class dog :public animal{
    public:
  void eat(){
       cout<<"eating bread"<<endl;
  }
};
int main(void)
{
    dog d=dog();
    d.animal::eat();

    return 0;
}
```

**[8]**

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

class A {
    int x = 5;
public:
    void display() {
        cout << "Value of x is: " << x << endl;
    }
};

class B : public A {
    int y = 10;
public:
    void display() {
        cout << "Value of y is: " << y << endl;
    }
};

int main() {
    A *a;
    B b;
    a = &b;
    a->display();
    return 0;
}
```