Develop python code to add, subtract , multiply and divide two distances where each distance contains two things of the format KM followed by Meters. (Example: d1 = 4km 500 m and d2 = 3 km 200 m)

In [49]:
```
class distance:

    def __init__(self,km,m):
        self.km=km
        self.m=m
    def display(self):
        print(self.km,"km",self.m,"m")
    def __add__(self,x):
        temp=distance(0,0)
        temp.km=self.km+x.km
        temp.m=self.m+x.m
        return temp
    def __sub__(self,x):
        temp=distance(0,0)
        temp.km=self.km-x.km
        temp.m=self.m-x.m
        return temp
    def __mul__(self,x):
        temp=distance(0,0)
        temp.km=self.km*x.km
        temp.m=self.m*x.m
        if (temp.m>=1000):
            temp.km=temp.km+(temp.m // 1000)
            temp.m=temp.m%1000
        return temp
    def __truediv__(self,x):
        temp=distance(0,0)
        temp.km=self.km/x.km
        temp.m=self.m/x.m
        return temp


d1=distance(4,500)
d2=distance(3,200)
print("First distance is :",end=" ")
d1.display()
print("Second distance is : ",end=" ")
d2.display()
d3=d1+d2
print("Total distance is : ",end=" ")
d3.display()

d3=d1-d2
print("Difference is  : ",end=" ")
d3.display()

d3=d1*d2
print("Product  is  : ",end=" ")
d3.display()

d3=d1/d2
print("Division is  : ",end=" ")
d3.display()
```

```
First distance is : 4 km 500 m
Second distance is :  3 km 200 m
Total distance is :  7 km 700 m
Difference is  :  1 km 300 m
Product  is  :  112 km 0 m
Division is  :  1.3333333333333333 km 2.5 m
```

Develop a python code to check given two dates d1 and d1 , check whether d1 is less than d2 or d1 is greater than d2 or d1 is equal to d2. (Hint: overload < , > , == operators)

In [50]:
```
class date:
    def __init__(self,d,m,y):
        self.d=d
        self.m=m
        self.y=y
    def display(self):
        print(self.d,"-",self.m,"-",self.y)

    def __lt__(self,x):
        if(self.d < x.d or self.m < x.m or self.y < x.y):
            return (True)
        else:
            return (False)

    def __gt__(self,x):
        if(self.d > x.d or self.m > x.m or self.y > x.y):
            return True

        else:
            return False


    def __eq__(self,x):
        if(self.d == x.d and self.m==x.m and self.y==x.y):
            return True

        else:
            return False



d1=date(10,10,2022)
d1.display()
d2=date(10,10,2022)
d2.display()

if (d1<d2):
       print("Less")
elif (d1>d2):
       print("Greater")
if d1==d2:
    print("Equal")
```

```
10 - 10 - 2022
10 - 10 - 2022
Equal
```

Develop a class called Box with attributes length, breadth, depth and define required constructor and other relevant methods. Inherit Box class to WeightBox which contains extra attribute as weight. From this extent further as ColorWeightBox which has Color as extra attribute. Develop code for entire scenario using multi-level inheritance.

In [35]:
```
class box:
    def __init__(self, l,b,d):
        self.l=l
        self.b=b
        self.d=d
    def display(self):
        print("length is : ",self.l,"cms")
        print("Breadth is : ",self.b,"cms")
        print("Depth is : ",self.d,"cms")

class weightbox(box):
    def __init__(self,l,b,d,w):
        box.__init__(self,l,b,d)
        self.w=w
```

```
    def display(self):
        box.display(self)
        print("Weight is : ",self.w,"kg")

class colorweightbox(weightbox):
    def __init__(self,l,b,d,w,c):
        weightbox.__init__(self,l,b,d,w)
        self.c=c

    def display(self):
        weightbox.display(self)
        print("Color is : ",self.c)

''''b=box(10,20,15)
b.display()
b1=weightbox(10,20,15,9)
b1.display() '''
b=colorweightbox(10,20,15,45,"red")
b.display()
```

```
length is :  10 cms
Breadth is :  20 cms
Depth is :  15 cms
Weight is :  45 kg
Color is :  red
```

Chef is a software developer, so he has to switch between different languages sometimes. Each programming language has some features, which are represented by integers here. Currently, Chef has to use a language with two given features A and B. He has two options --- switching to a language with two features A1 and B1, or to a language with two features A2 and B2. All four features of these two languages are pairwise distinct. Tell Chef whether he can use the first language, the second language or neither of these languages (if no single language has all the required features) The first and only line of each test case contains six space-separated integers A,B,A1,B1,A2,B2. For each test case, print a single line containing the integer 1 if Chef should switch to the first language, or 2 if Chef should switch to the second language, or 0 if Chef cannot switch to either language.

In [ ]:
```
n=int(input())
for i in range(n):
    a,b,a1,b1,a2,b2=map(int,input().split())
    l1=[]
    l2=[]
    l3=[]

    l1.append(a)
    l1.append(b)
    l2.append(a1)
    l2.append(b1)
    l3.append(a2)
    l3.append(b2)

    l1.sort()
    l2.sort()
    l3.sort()



    if l1==l2:
        print("1")
    elif l1==l3:
        print("2")
    else:
        print("0")
```

You have prepared four problems. The difficulty levels of the problems are A1,A2,A3,A4 respectively. A

problem set comprises two problems and no two problems in a problem set should have the same difficulty level. A problem can belong to at most one problem set. Find the maximum number of problem sets you can create using the four problems. Each test case contains four space-separated integers A1, A2, A3, A4, denoting the difficulty level of four problems. For each test case, print a single line containing one integer - the maximum number of problem sets you can create using the four problems.

In [ ]:
```
n=int(input())
for i in range(n):
    a1,a2,a3,a4=map(int,input().split())
    if a1!=a2 and a3!=a4 and a2!=a3 and a1!=a4 and a2!=a4 and a1!=a3:
        print("2 sets")
    elif a1==a2==a3==a4:
        print("0 sets")
    else:
        print("1 set")
```