```
1.z= int(input())
for i in range(z):
    A,B,A1,B1,A2,B2=map(int,input().split())
    if (A1==A or A1==B) and (B1==B or B1==A):
        print("1")
    elif (A2==A or A2==B) and (B2==B or B2==A):
        print("2")
    else:
        print("0")

2.t=int(input())
l=[]
s=set()
while(t>0):
    for i in range(0,4):
        i=int(input())
        s.add(i)
        l.append(i)
    #print(len(s))
    if(len(s)==4):
        print("2")
    elif(len(s)==1):
        print("0")
    elif(len(s)==3):
        print("1")
    elif(len(s)==2):
        b=0
        c=l[0]
        for i in range(0,3):
            if(c==l[i+1]):
                b=b+1
                #print(b)
        if(b==1):
            print("2")
        else:
            print("1")
    s.clear()
    l.clear()
    t=t-1
3.1.class Date:
    def __init__(self, day, month, year):
        self.day = day
        self.month = month
        self.year = year

    def __lt__(self, other):
        if self.year < other.year:
            return True
        elif self.year == other.year and self.month < other.month:
            return True
```

```
        elif self.year == other.year and self.month == other.month and self.day <
other.day:
            return True
        else:
            return False

    def __gt__(self, other):
        if self.year > other.year:
            return True
        elif self.year == other.year and self.month > other.month:
            return True
        elif self.year == other.year and self.month == other.month and self.day >
other.day:
            return True
        else:
            return False

    def __eq__(self, other):
        return (
            self.year == other.year and
            self.month == other.month and
            self.day == other.day
        )


if d1 < d2:
    print("d1 is less than d2")
elif d1 > d2:
    print("d1 is greater than d2")
else:
    print("d1 is equal to d2")

3.2.
class Distance:
    def __init__(self, km, meters):
        self.km = km
        self.meters = meters

    def __str__(self):
        return f"{self.km}KM {self.meters}M"

    def __add__(self, other):
        total_meters = self.to_meters() + other.to_meters()
        return Distance.from_meters(total_meters)

    def __sub__(self, other):
        total_meters = self.to_meters() - other.to_meters()
        return Distance.from_meters(total_meters)

    def __mul__(self, other):
```

```
        total_meters = self.to_meters() * other
        return Distance.from_meters(total_meters)

    def __truediv__(self, other):
        total_meters = self.to_meters() / other
        return Distance.from_meters(total_meters)

    def to_meters(self):
        return self.km * 1000 + self.meters

    @staticmethod
    def from_meters(total_meters):
        km = total_meters // 1000
        meters = total_meters % 1000
        return Distance(km, meters)

4.class Box:
    def __init__(self, length, breadth, depth):
        self.length = length
        self.breadth = breadth
        self.depth = depth

    def display_dimensions(self):
        print(f"Dimensions: {self.length} x {self.breadth} x {self.depth}")


class WeightBox(Box):
    def __init__(self, length, breadth, depth, weight):
        super().__init__(length, breadth, depth)
        self.weight = weight

    def display_weight(self):
        print(f"Weight: {self.weight} kg")


class ColorWeightBox(WeightBox):
    def __init__(self, length, breadth, depth, weight, color):
        super().__init__(length, breadth, depth, weight)
        self.color = color

    def display_color(self):
        print(f"Color: {self.color}")
```