Assignment

1. chef is a software developer, so he has to switch between different language 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 A_2 and B_2. All four of these 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, A2, B1, B_2. 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.

Program :

```
n= int(input())
for i in range n :
    a,b,a1,b1,a2,b2 = tuple(map(int, input().split()))
    if(a==a1 and b==b1) or (a==b1 and b==a1) :
        print (1)
    elif (a==a2 and b== b2) or (a==b2 and b==a2) :
        print(2)
    else :
        print(0)
```

1. 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 same difficulty level. A problem can belong to atmost 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 four problems.

<u>Program</u> :

```
T= int(input())
for i in range (T):
    l = list (map (int, input (). split())
    a= set(l)
    if (len(a) == 4):
        print(2)
    elif (len(a) == 3):
        print (2)
    elif (len(a) == 2):
        l.sort()
        b=l[0]
        if (l.count (b) == 2) :
```

```
        print(2)
    else :
        print(1)

elsef :
    print(0)
```

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

```
from datetime import date
class mydate():
  def_init_(self, date1):
    Self.date1 = date1

  def_gt_(self, date2):
    return self.date1 > date2
  def_lt_(self, date2):
    return self.date1< date2
  def_eq_(self, date2):
    return self.date1 = = date2
x = date(2013, 2, 5)
c = mydate(x)
y = date(2013, 2, 2)
def fun(c, y):
  if (c._gt_(y)):
    return "greater than "
    if(c.__lt__(y)):
        return "less than"
    if (c.__eq__(y)):
        return "equal"
print(fun(c,y))
```

2. Develop a python code to add, subtract, multiply & divide two distances where each distance contains two things of the format KM followed by metres (Ex: d_1 = 4 km 500m and d_2 = 3 km 200m).

```
d1 = input().split()
d2 = input().split()
m1 = int (d1[0][:-2]) * 1000 + int (d1[1][:-1])
m2 = int (d2[0][:-2]) * 1000 + int (d1[1][:-1])
def add (m1, m2):
    k = (m1 + m2) //1000
    m = (m1 + m2) %1000
    return f"{k}km {m}m"
def sub (m1, m2):
    k = (m1 - m2)//1000
    m = (m1 - m2) %1000
    return f"{k} km {m}m"
def mul (m1, m2):
    k = (m1 * m2) //1000
    m = (m1 - m2) %1000
    return f"{k}km {m}m"
```

```
def div (m1 ,m2) :
    k = (m1/m2) //1000
    m = (m1/m2) %.1000
    return f" {k} km {m}m"
```

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

```
class Box :
    def __init__ (self, length, breadth, depth):
        self.length = length
        self. breadth = breadth
        self. depth = depth.
    def area (self):
        return self.length * self. breadth .

    def volume(self):
        return self.length * self.breadth * self.depth

    def perimeter (self):
        return 2*(length + breadth)

class Weight Box (Box):
    def __init__(self, weight):
        Self.weight = weight
    def getWeight (self):
        return self.weight
    def setweight (self):
        ~~return~~
        self.weight = weight
class ColorWeightBox (WeightBox):
    def __init__(self, color):
        self.color = color
    def getcolor (self):
        return self.color
    def setColor (self, color):
        self.color = color
```