'''Chef is a software developer, so he has to switch between different languages sometimes. Each programming langu age 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 featu res)
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 th e second language, or 0 if Chef cannot switch to either language.'''

```
def selectLanguauge():

    a=int(input("Enter A value: "))
    b=int(input("Enter B value: "))
    a1=int(input("Enter A1 value: "))
    b1=int(input("Enter B1 value: "))
    a2=int(input("Enter A2 value: "))
    b2=int(input("Enter B2 value: "))
    if(a > b):
        a, b = b, a
    if(a1 > b1):
        a1, b1 = b1, a1
    if(a2 > b2):
        a2, b2 = b2, a2

    if(a == a1 and b == b1):
        return 1
    elif(a == a2 and b == b2):
        return 2
    else:
        return 0

selectLanguauge()
```

'''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 pro blems.
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 u sing the four problems.'''

```
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)
    else:
        print(0)
```

'''Develop a python code to check given two dates d1 and d1 , check whether d1 is less than d2 or d1 is greater than d 2 or d1 is equal to d2. (Hint: overload < , > , == operators)'''

```
from datetime import *

# Enter first date
d1, m1, y1 = [int(x) for x in input("Enter first"
        " person's date(DD/MM/YYYY) : ").split('/')]

b1 = date(y1, m1, d1)

# Enter second date
d2, m2, y2 = [int(x) for x in input("Enter second"
        " person's date(DD/MM/YYYY) : ").split('/')]

b2 = date(y2, m2, d2)

# Comparing the dates will return
# either True or False
print("date1 is greater than date2 : ", b1 > b2)
print("date1 is less than date2 : ", b1 < b2)
print("date1 is not equal to date2 : ", b1 != b2)
```

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

```
class Distance:
    def GetDistance(self):
        #self.__cm=int(input("Enter CM: "))
        self.__m=int(input("Enter M: "))
        self.__km = int(input("Enter KM: "))

    def PutDistance(self):
        print(self.__km,self.__m)

    def __add__(self, T):
        R=Distance()
        R.__m=self.__m+T.__m
        R.__km = self.__km + T.__km
        R.__km=R.__km+(R.__m//1000)
        R.__m=R.__m%1000

        return R

D1=Distance()
D2=Distance()
```

```
print("Enter first distance")
D1.GetDistance()

print("Enter second distance")
D2.GetDistance()

D3=D1+D2

print("The sum of both distance is")
D3.PutDistance()
```

'''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 Color WeightBox which has Color as extra attribute. Develop code for entire scenario using multi-level inheritance.'''

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

class WeightBox(Box):
    def __init__(self,weight):
         self.weight = weight

    def getCapacity(self, sBox):
            return self.weight * sBox.getVolume()

class ColorWeightBox(WeightBox):
    def __init__(self,color):
         self.color = color

    def getBoxInfo(self, sWeightBox, sBox):
            capacity = sWeightBox.getCapacity(sBox)
            return "Box is in " + self.color + " color and its capacity is : " + str(capacity)


smallBox = Box(1,1,3)
weightBox = WeightBox (4)
colorWeightBox = ColorWeightBox ("Pink")
result = colorWeightBox.getBoxInfo(weightBox, smallBox)
print(result)
```