Assignment 6

1. 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.

Solution:

```
#include <bits/stdc++.h>


using namespace std;


typedef long long ll;

#define forn(i,e) for(ll i = 0; i < e; i++)


void solve()

{

   ll a,b,a1,b1,a2,b2;

   cin>>a>>b>>a1>>b1>>a2>>b2;

   if(a > b) swap(a,b);

   if(a1 > b1) swap(a1,b1);

   if(a2 > b2) swap(a2,b2);


   if(a == a1 && b == b1)

   {

      cout<<1<<endl;

   }

   else if(a == a2 && b == b2)
```

```
    {
        cout<<2<<endl;
    }
    else
    {
        cout<<0<<endl;
    }
}

int main()
{
            ll t=1;
            cin >> t;
    forn(i,t) {
            solve();
    }
    return 0;
}
```

2. 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

Solution:

```
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)
```

3. 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.

Solution :

```
import datetime

# date and time in yyyy/mm/dd hh:mm:ss format

d1 = datetime.datetime(2020, 5, 11, 22, 50, 55)

d2 = datetime.datetime(2020, 7, 11, 22, 50, 55)

print(d1 < d2)

print(d1 > d2)

print(d1 == d2)
```

4. 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)

Solution:

```
class Distance:

  def GetDistance(self):

    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

    def __sub__(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

    def __mult__(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

    def __div__(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 add of both distance is")

D3.PutDistance()


D4 = D1-D2

print("The sub of both distance is")

D4.PutDistance()


D5 = D1 * D2

print("The mult of both distance is")

D5.PutDistance()


D6 = D1/D2

print("The div of both distance is")

D6.PutDistance()
```

5. 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.

```
class Box():

   def  base_parameters(self):
        length = 10
        bredth = 20
        width = 30
        print("L:B:W",length,bredth,width)

class WeightBox(Box):
   def extra_feature(self):
        weight_kgs = 100
        print("weight in kgs",weight_kgs)

class ColorWeightBox(WeightBox):
   def color(self):
        color = "blue"
        print("Color of the cube",color)

call = ColorWeightBox()
call.base_parameters()
call.extra_feature()
call.color()
```