Assignment_06
Submitted By: UmaPavan Kumar Kethavarapu.

1.Chef is a software developer, so he must switch between different languages sometimes. Each programming language has some features, which are represented by integers here. Currently, Chef must 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.
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.

A and B A --> A1 and B1 B--> A2 and B2 Read A,B along with A1 , B1 and A2,B2-->six integers if A is 1 then print A if B is 1 then B is 1

In [5]:

```python
A,B,A1,B1,A2,B2 = map(int,input("Enter 6 Values").split())
N =0
if A==1 and B==0:
    print ('Selected A language',A)
elif A==0 and B==1:
    print('Selected B Language',B)
else:
    print('Nither Langauge Selected',N)
```

```
Enter 6 Values1 0 0 1 1 0
Selected A language 1
```

In [6]:

```python
A,B,A1,B1,A2,B2 = map(int,input("Enter 6 Values").split())
N =0
if A==1 and B==0:
    print ('Selected A language',A)
elif A==0 and B==1:
    print('Selected B Language',B)
else:
    print('Nither Langauge Selected',N)
```

```
Enter 6 Values0 1 1 0 0 1
Selected B Language 1
```

In [7]:

```
A,B,A1,B1,A2,B2 = map(int,input("Enter 6 Values").split())
N =0
if A==1 and B==0:
    print ('Selected A language',A)
elif A==0 and B==1:
    print('Selected B Language',B)
else:
    print('Nither Langauge Selected',N)
```

```
Enter 6 Values0 0 0 0 1 1
Nither Langauge Selected 0
```

In [10]:

```
    a = list(map(int,input('Enter 4 Values').split()))
    b = set(a)
    if len(b)==4:
        print(2)
    elif len(b)==3 or len(b)==2:
        print(1)
    else:
        print(0)
```

```
Enter 4 Values1 2 3 4
2
```

In [12]:

```
    a = list(map(int,input('Enter 4 Values').split()))
    b = set(a)
    if len(b)==4:
        print(2)
    elif len(b)==3 or len(b)==2:
        print(1)
    else:
        print(0)
```

```
Enter 4 Values1 2 1 2
1
```

In [13]:

```
    a = list(map(int,input('Enter 4 Values').split()))
    b = set(a)
    if len(b)==4:
        print(2)
    elif len(b)==3 or len(b)==2:
        print(1)
    else:
        print(0)
```

```
Enter 4 Values1 2 3 1
1
```

In [14]:

```
    a = list(map(int,input('Enter 4 Values').split()))
    b = set(a)
    if len(b)==4:
        print(2)
    elif len(b)==3 or len(b)==2:
        print(1)
    else:
        print(0)
```

```
Enter 4 Values1 1 1 1
0
```

a. 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 [61]:

```
class dat:
    def __init__(self,day,month,year):
        self.day = day
        self.month = month
        self.year = year
    def __eq__(self,other):
        if self.day==other.day and self.month ==other.month and self.year ==other.year:
            print('Dates Are Equal')
        else:
            print('Date Are Not Equal')
    def __ge__(self,other):
        if self.day>=other.day and self.month>=other.month and self.year>=other.year:
            print('Date1 Is > Date2')
        else:
            print("Date2 Is > Date1")
    def __le__(self,other):
        if self.day<=other.day and self.month<=other.month and self.year<=other.year:
            print('Date1 Is < Date2')
        else:
            print("Date2 Is < Date1")
```

In [62]:

```
d1 = dat(10,10,2022)
```

In [63]:

```
d2 = dat(10,10,2022)
```

In [64]:

```
print(d1==d2)
```

```
Dates Are Equal
None
```

In [65]:

```
d1 = dat(10,10,2021)
d2 =dat(10,10,2022)
```

In [66]:

```
print(d1==d2)
```

```
Date Are Not Equal
None
```

In [67]:

```
d1 = dat(10,10,2021)
d2 =dat(10,10,2022)
```

In [68]:

```
print(d1>=d2)
```

```
Date2 Is > Date1
None
```

In [69]:

```
d1 = dat(10,10,2021)
d2 =dat(10,10,2022)
```

In [70]:

```
print(d1<=d2)
```

```
Date1 Is < Date2
None
```

b. Develop python code to add, subtract, multiply and divide two distances where each distance contains two things of the format KM followed by Meters.

Test Case 1:

In [16]:

```
dict1 = {'KM':100,'Meters':45}
dict2 = {'KM1':200,'Meters1':55}
```

In [17]:

```
dict1
```

Out[17]:

```
{'KM': 100, 'Meters': 45}
```

In [18]:

```
dict2
```

Out[18]:

```
{'KM1': 200, 'Meters1': 55}
```

In [27]:

```
list1=list(dict1.values())
list2=list(dict2.values())
item1=list1[0]+list2[0]
item2=list1[1]+list2[1]
print('Kilo Meters Addition:',item1)
print('Meters Addition:',item2)
```

```
Kilo Meters Addition: 300
Meters Addition: 100
```

Test Case:2

In [28]:

```
dict1 = {'KM':4,'Meters':500}
dict2 = {'KM1':3,'Meters1':200}
```

In [29]:

```
list1=list(dict1.values())
list2=list(dict2.values())
item1=list1[0]+list2[0]
item2=list1[1]+list2[1]
print('Kilo Meters Addition:',item1)
print('Meters Addition:',item2)
```

```
Kilo Meters Addition: 7
Meters Addition: 700
```

2. 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 [73]:

```
class Box:
    def __init__(self,l,b,d):
        self.l = l
        self.b = b
        self.d = d
    def disp(self):
        print('Length:',self.l)
        print('Breadth:',self.b)
        print('Depth:',self.d)
class WeightBox(Box):
    def __init__(self,l,b,d,w):
        self.w = w
        Box.__init__(self,l,b,d)
    def disp(self):
        print('Length:',self.l)
        print('Breadth:',self.b)
        print('Depth:',self.d)
        print('Weight:',self.w)
class ColorWeightBox(WeightBox):
    def __init__(self,l,b,d,w,c):
        self.c = c
        WeightBox.__init__(self,l,b,d,w)
    def disp(self):
        print('Length:',self.l)
        print('Breadth:',self.b)
        print('Depth:',self.d)
        print('Weight:',self.w)
        print('Color:',self.c)
```

In [74]:

```
obj = ColorWeightBox(10,20,30,100,'Green')
obj.disp()
```

```
Length: 10
Breadth: 20
Depth: 30
Weight: 100
Color: Green
```