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.

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 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.

1. 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)
2. 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)

1. 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.

Ans:

1. Chef's Programming Language:

```
def check_language(A, B, A1, B1, A2, B2):
   if (A == A1 and B == B1) or (A == A2 and B == B2):
      return 1
   elif (A == A1 and B == B2) or (A == A2 and B == B1):
      return 2
   else:
      return 0

# Example usage
result = check_language(1, 2, 3, 4, 5, 6)
print(result)
```

2. Maximum Number of Problem Sets:

```
def max_problem_sets(A1, A2, A3, A4):
```

```
    difficulties = [A1, A2, A3, A4]
    max_sets = min(len(set(difficulties)), 2)
    return max_sets

# Example usage
result = max_problem_sets(1, 2, 3, 4)
print(result)
```

3. Comparison of Dates:

```
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 self.year < other.year
        elif self.month != other.month:
            return self.month < other.month
        else:
            return self.day < other.day

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

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

# Example usage
date1 = Date(10, 5, 2023)
date2 = Date(20, 5, 2023)

if date1 < date2:
    print("d1 is less than d2")
elif date1 > date2:
    print("d1 is greater than d2")
else:
    print("d1 is equal to d2")
```

4. Distance Calculation:

```
class Distance:
```

```
def __init__(self, km, meters):
    self.km = km
    self.meters = meters

def __add__(self, other):
    total_km = self.km + other.km
    total_meters = self.meters + other.meters

    if total_meters >= 1000:
        total_km += 1
        total_meters -= 1000

    return Distance(total_km, total_meters)

def __sub__(self, other):
    if self < other:
        raise ValueError("Subtraction results in negative distance")

    total_km = self.km - other.km
    total_meters = self.meters - other.meters

    if total_meters < 0:
        total_km -= 1
        total_meters += 1000

    return Distance(total_km, total_meters)

def __mul__(self, scalar):
    total_meters = (self.km * 1000 + self.meters) * scalar
    total_km = total_meters // 1000
    total_meters %= 1000

    return Distance(total_km, total_meters)

def __truediv__(self, scalar):
    total_meters = (self.km * 1000 + self.m
```