41. Date Format

Write a program to read two String variables in DD-MM-YYYY.Compare the two dates and return the older date in 'MM/DD/YYYY' format.

Include a class UserMainCode with a static method **findOldDate** which accepts the string values. The return type is the string.
Create a Class Main which would be used to accept the two string values and call the static method present in UserMainCode.

Input and Output Format:
Input consists of two string.
Output consists of a string.
Refer sample output for formatting specifications.

Sample Input 1:
05-12-1987
8-11-2010
Sample Output 1:
12/05/1987

Solution:

```
import java.text.ParseException;
import java.util.Scanner;

public class Main {
	public static void main(String[] args) throws ParseException {
		Scanner s=new Scanner(System.in);
		String s1=s.next();
		String s2=s.next();
		System.out.println(User.findOldDate(s1,s2));
	}

}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;

public class User {
	public static String findOldDate(String s1,String s2) throws ParseException
	{
		SimpleDateFormat sd1=new SimpleDateFormat("dd-MM-yyyy");
		Date d1=sd1.parse(s1);
		Date d2=sd1.parse(s2);
```

```
            Calendar c=Calendar.getInstance();
            c.setTime(d1);
            int day1=c.get(Calendar.DAY_OF_MONTH);
            int m1=c.get(Calendar.MONTH);
            int y1=c.get(Calendar.YEAR);
            c.setTime(d2);
            int day2=c.get(Calendar.DAY_OF_MONTH);
            int m2=c.get(Calendar.MONTH);
            int y2=c.get(Calendar.YEAR);
            SimpleDateFormat sd2=new SimpleDateFormat("MM/dd/yyyy");
            String res=null;
            if(y1==y2)
            {
                  if(m1==m2)
                  {
                        if(day1==day2)
                        {
                              res=sd2.format(d1);
                        }
                  }
                        else
                        {
                              if(m1>m2)
                                    res=sd2.format(d2);
                              else
                                    res=sd2.format(d1);
                        }
                  }
            else
            {
                  if(y1>y2)
                        res=sd2.format(d2);
                  else
                        res=sd2.format(d1);
            }

            return res;
      }
}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;

import java.util.Date;
public class Palindrome {
            public static String removeDuplicate(String s1,String s2) throws
ParseException
            {
                  SimpleDateFormat sd1=new SimpleDateFormat("dd-MM-yyyy");
```

```
                    Date d1=sd1.parse(s1);
                    Date d2=sd1.parse(s2);
                    String res=null;

                    SimpleDateFormat sfd2=new SimpleDateFormat("MM/dd/yyyy");


                    if(d1.compareTo(d2)<0)
                    {
                          res=sfd2.format(d1);
                    }
                    else
                    {
                          res=sfd2.format(d2);
                    }

                    return res;

                    }
}
```

42. Interest calculation

1. Read account details from the User. The details would include id, DOB (date of birth) and amount in the given order. The datatype for id is string, DOB is string and amount is integer.
2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOB as value, and the second hashmap contains same employee ids as key and amount as value.
3. Rate of interest as on 01/01/2015:
   a. If the age greater than or equal to 60 then interest rate is 10% of Amount.
   b.If the age less then to 60 and greater than or equal to 30 then interest rate is 7% of Amount.
   v. If the age less then to 30 interest rate is 4% of Amount.
4. Revised Amount= principle Amount + interest rate.
5. You decide to write a function **calculateInterestRate** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.

Input and Output Format:
Input consists of account details. The first number indicates the size of the acoount. The next three values indicate the user id, DOB and amount. The Employee DOB format is “dd-mm-yyyy”
Output consists of the user id and the amount for each user one in a line.
Refer sample output for formatting specifications.

Sample Input 1:
4
SBI-1010
20-01-1987
10000
SBI-1011
03-08-1980
15000
SBI-1012
05-11-1975
20000
SBI-1013
02-12-1950
30000
Sample Output 1:
SBI-1010:10400
SBI-1011:16050
SBI-1012:21400
SBI-1013:33000

43. Discount rate calculation

Write a program to calculate discount of the acccount holders based on the transaction amount and registration date using below mentioned prototype:
1. Read account details from the User. The details would include id, DOR (date of registration) and transaction amount in the given order. The datatype for id is string, DOR is string and transaction amount is integer.
2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOR as value, and the second hashmap contains same employee ids as key and amount as value.
3. Discount Amount as on 01/01/2015:
   a. If the transaction amount greater than or equal to 20000 and registration greater than or equal to 5 year then discount rate is 20% of transaction amount.
   b. If the transaction amount greater than or equal to 20000 and registration less then to 5 year then discount rate is 10% of transaction amount.
   c. If the transaction amount less than to 20000 and registration greater than or equal to 5 year then discount rate is 15% of transaction amount.
   d. If the transaction amount less than to 20000 and registration less then to 5 year then discount rate is 5% of transaction amount.
4. You decide to write a function **calculateDiscount** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.

Input and Output Format:

Input consists of transaction details. The first number indicates the size of the employees. The next three values indicate the user id, user DOR and transaction amount. The DOR (Date of Registration) format is “dd-mm-yyyy”

Output consists of a string which has the user id and discount amount one in a line for each user. Refer sample output for formatting specifications.

Sample Input 1:

4
A-1010
20-11-2007
25000
B-1011
04-12-2010
30000
C-1012
11-11-2005
15000
D-1013
02-12-2012
10000

Sample Output 1:

A-1010:5000
B-1011:3000
C-1012:2250
D-1013:500

Solution:

```
public class main {
public static void main(String []args){
Scanner sc=new Scanner(System.in);
int s=Integer.parseInt(sc.nextLine());
HashMap<String,String>hm=new HashMap<String,String>();
HashMap<String,Integer>hm1=new HashMap<String,Integer>();
for(int i=0;i<s;i++)
{
String id=sc.nextLine();
hm.put(id, sc.nextLine());
hm1.put(id,Integer.parseInt(sc.nextLine()));
}
```

```
TreeMap<String,Integer>tm=new TreeMap<String,Integer>();
tm=Usermaincode.findDiscountRate(hm,hm1);
Iterator<String> it=tm.keySet().iterator();
while(it.hasNext())
{
String n=it.next();
int fac=tm.get(n);
System.out.println(n+":"+fac);
}
}
```

```
public class UserMaincode
{
public static TreeMap<String,Integer> findDiscountRate
(HashMap<String,String>hm,HashMap<String,Integer>hm1) throws ParseException
{
TreeMap<String,Integer> tm=new TreeMap<String,Integer>();
SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
Iterator<String> itr1=hm.keySet().iterator();
while(itr1.hasNext())
{
        try
        {
        String id=itr1.next();
        String dor=hm.get(id);
        int am=hm1.get(id);
        Date d1=sdf.parse(dor);
        String s1="01-01-2015";
        Date d2=sdf.parse(s1);
        int y1=d1.getYear();
        int m1=d1.getMonth();
        int day1=d1.getDay();
        int y2=d2.getYear();
        int m2=d2.getMonth();
        int day2=d2.getDay();
        int exp=Math.abs(y1-y2);
        if(m1==m2)
        {
                if(day2>day1)
                        exp--;
                }
```

```
		if(m2>m1)
			exp--;
		if(am>=20000 && exp>=5)
		{
			int dis=(int) (0.20*am);
			tm.put(id,dis);
		}
		else if(am>=20000 && exp<5)
		{
			int dis=(int) (0.1*am);
			tm.put(id,dis);
		}
		else if(am<20000 && exp>=5)
		{
			int dis=(int) (0.15*am);
			tm.put(id,dis);
		}
		else if(am<20000 && exp<5)
		{
			int dis=(int) (0.05*am);
			tm.put(id,dis);
		}
		}
		catch(Exception e){
			System.out.println(e);
		}
}
return tm;
}
}
}
```