```
import java.io.IOException;
import java.io.StringWriter;
import java.util.ArrayList;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class People_You_May_Know {
//Used in Reduce Function
public static int NumberOfUsersInSample = 0;
public static int ReducerIterationCount = 0;


public static class Map
extends Mapper < LongWritable, Text, Text, IntWritable >
{

private final static IntWritable USER = new IntWritable( 1);
private Text FRIEND = new Text();

@Override
public void map( LongWritable key, Text value, Context context
) throws IOException,
InterruptedException
{
   //map output is generating all k-v pairs (all friend pairs possible)
   String UserId = "";
   ArrayList<String> FriendIds = new ArrayList<String>();
   //split with Tab
   String temp1[] = String.valueOf(value).split("\t");
   UserId = temp1[0];
   int N = temp1.length;
   if(N==2)
   {
      NumberOfUsersInSample++;
      //Storing FriendIds
      String temp2[] = String.valueOf(temp1[1]).split(",");
      int NumberOfFriends = temp2.length;
```

```
      for(int a=0;a<NumberOfFriends;a++)
      {
         FriendIds.add(temp2[a]);
      }
      //user to friends
      for(int b=0;b<NumberOfFriends;b++)
      {
         //send key value pairs to reducer
         USER.set(Integer.valueOf(UserId));
         FRIEND.set(FriendIds.get(b));
         System.out.println("USER: "+USER);
         System.out.println("FRIEND: "+FRIEND);
         context.write( FRIEND,USER);
      }
      //friends to user
      for(int b=0;b<NumberOfFriends;b++)
      {
         //send key value pairs to reducer
         USER.set(Integer.valueOf(FriendIds.get(b)));
         FRIEND.set(UserId);
         System.out.println("USER: "+USER);
         System.out.println("FRIEND: "+FRIEND);
         context.write( FRIEND,USER);
      }
   }
}
}
public static class Reduce
extends
Reducer < Text, IntWritable, Text, Text > {
private Text result = new Text();
public ArrayList<String> K = new ArrayList<String>();
@Override
public void reduce( Text key, Iterable < IntWritable > values, Context context
) throws IOException,
InterruptedException {
   //Find all final friend lists
   ArrayList<Integer> V = new ArrayList<Integer>();
   for (IntWritable val : values)
   {
      V.add(val.get());
   }
   ReducerIterationCount++;
   System.out.println("V: "+key+" : "+V);
   String temp = String.valueOf(V);
   temp = temp.replace("[", "");
   temp = temp.replace("]", "");
   K.add(" "+String.valueOf(key)+" :"+temp);
```

```
   String ActualUser = "";
   System.out.println("ReducerIterationCount: "+ReducerIterationCount);
   System.out.println("NumberOfUsersInSample: "+NumberOfUsersInSample);
   if(ReducerIterationCount==NumberOfUsersInSample)
   {
      for(int d=0;d<K.size();d++)
      {
         //mutual friends list
         ArrayList<Integer> W = new ArrayList<Integer>();
         String str = K.get(d);
         String S[] = str.split(":");
         //Given Users
         if(S[0].contains(" 150 ")||S[0].contains(" 8942 ")||S[0].contains(" 6157 ")||S[0].contains(" 9019
")||S[0].contains(" 9020 ")||S[0].contains(" 9021 ")||S[0].contains(" 9022 ")||S[0].contains(" 9990
")||S[0].contains(" 9992 ")||S[0].contains(" 9993 "))
         {
            ActualUser = S[0].trim();
            String T[] = S[1].split(",");
            //collect mutual friends
            for(int e=0;e<T.length;e++)
            {
               for(int f=0;f<K.size();f++)
               {
                  String X = T[e].trim();
                  String Y = K.get(f);
                  String Z[] = Y.split(":");
                  if((Z[0].trim()).equalsIgnoreCase(X.trim()))
                  {
                     String M[] = Z[1].split(",");
                     //adding mutual friends
                     for(int g=0;g<M.length;g++)
                     {
                        W.add(Integer.parseInt(M[g].trim()));
                     }
                  }
               }
            }
            String R = String.valueOf(W);
            //remove actual user
            if(R.contains(" "+ActualUser+","))
               R = R.replaceAll(" "+ActualUser+",", "");
            if(R.contains("["+ActualUser+", "))
               R = R.replace("["+ActualUser+", ", "");
            if(R.contains(", "+ActualUser+"]"))
               R = R.replace(", "+ActualUser+"]", "");
            //remove friends from mutual friends list
            for(int h=0;h<T.length;h++)
            {
```

```
   String N = T[h].trim();
   if(R.contains(" "+N+","))
      R = R.replaceAll(" "+N+",", "");
   if(R.contains("["+N+", "))
      R = R.replace("["+N+", ", "");
   if(R.contains(", "+N+"]"))
      R = R.replace(", "+N+"]", "");
}
R = R.replace("[", "");
R = R.replace("]", "");
//Final Mutual Friends List
String MFLtemp[] = R.split(",");
int MFLlength = MFLtemp.length;
//Find mode,add to final list and remove from mutual friends list
int looplength = Math.min(MFLlength, 10);
System.out.println("looplength: "+looplength);
String Res = "";
for(int q =0;q<looplength;q++)
{
   int mode1 = 0,mode2 =0,c1=0,c2=0;
   String MFL[] = R.split(",");
   MFLlength = MFL.length;
   System.out.println("MFLlength: "+MFLlength);
   if(MFLlength==1)
      break;
   for(int o=0;o<MFLlength;o++)
   {
      mode1 = Integer.parseInt(MFL[o].trim());
      c1 = 1;
      for(int p=o+1;p<MFLlength;p++)
      {
         if(mode1==Integer.parseInt(MFL[p].trim()))
            c1++;
      }
      if(c1>c2)
      {
         mode2 = mode1;
         c2=c1;
      }
      else if(c1==c2)
      {
         mode2 = Math.min(mode1, mode2);
      }
   }
   Res = Res+String.valueOf(mode2)+",";
   System.out.println("Res: "+Res);
   //remove the mode - for next PYMK value
   R = "["+R+"]";
```

```
            if(R.contains(" "+String.valueOf(mode2)+","))
               R = R.replaceAll(" "+String.valueOf(mode2)+",", "");
            if(R.contains("["+String.valueOf(mode2)+", "))
               R = R.replace("["+String.valueOf(mode2)+", ", "");
            if(R.contains(", "+String.valueOf(mode2)+"]"))
               R = R.replace(", "+String.valueOf(mode2)+"]", "");
            R = R.replace("[", "");
            R = R.replace("]", "");
          }
          Res = Res.substring(0, Res.length()-1);
          System.out.println("ActualUser: "+ActualUser);
          System.out.println("Res: "+Res);
          context.write(new Text(ActualUser), new Text(Res));
        }
      }
    }
}
}

public static void main( String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = Job.getInstance( conf, "PYMK");

job.setJarByClass(People_You_May_Know.class);
FileInputFormat.addInputPath( job, new Path("input"));
FileOutputFormat.setOutputPath( job, new Path("output"));

job.setMapperClass( Map.class);
//job.setCombinerClass( Reduce.class);
job.setReducerClass( Reduce.class);

job.setOutputKeyClass( Text.class);
job.setOutputValueClass( IntWritable.class);

System.exit( job.waitForCompletion( true) ? 0 : 1);
}
}
```