

```

/**
 * American Computer Science League Contest #1: Intermediate Division
 */
package intermediate;

import java.io.*; // package contains BufferedReader class
import java.util.*; // package contains StringTokenizer class

/**
 * Given the length sides of two
 * @author gdonaldson
 */
class Triangles // Intermediate Division
{
    public static void main( String[] args ) throws IOException
    {
        // Use BufferedReader to capture data from data file on disk drive
        BufferedReader in = new BufferedReader(
            new FileReader( "TrianglesInter.in" ) );

        // == Read five lines containing six double tokens per line line. ==

        double AB=0, BC=0, AC=0, DE=0, EF=0, DF=0 ;

        for( int i = 0; i < 5; i++ )
        {
            StringTokenizer st = new StringTokenizer( in.readLine() );
            AB = Double.parseDouble( st.nextToken() );
            BC = Double.parseDouble( st.nextToken() );
            AC = Double.parseDouble( st.nextToken() );
            DE = Double.parseDouble( st.nextToken() );
            EF = Double.parseDouble( st.nextToken() );
            DF = Double.parseDouble( st.nextToken() );

            if ( AB==DE && BC==EF && AC==DF) System.out.println("DEF");

            else if ( AB==DE && BC==DF && AC==EF) System.out.println("EDF");

            else if ( AB==EF && BC==DE && AC==DF) System.out.println("FED");
            else if ( AB==EF && BC==DF && AC==DE) System.out.println("EFD");

            else if ( AB==DF && BC==DE && AC==EF) System.out.println("FDE");

            else if ( AB==DF && BC==EF && AC==DE) System.out.println("DFE");

            else System.out.println("NONE");

        } // end for

        System.exit( 0 ); // don't omit this

    } // end main()
} // end class Triangles [ Intermediate Division ]

```

```

/**
 * American Computer Science League Contest #1: Senior Division
 */
package senior;

import java.io.*; // package contains BufferedReader class
import java.util.*; // package contains StringTokenizer class

class Triangles
{
    private static double AB = 0, BC = 0, AC = 0, DE = 0, EF = 0, DF = 0;

    public static void main( String[] args ) throws IOException
    {
        // Use BufferedReader to capture data from data file on disk drive
        BufferedReader in = new BufferedReader(
            new FileReader( "TestSr.in" ) );

        for( int i = 0; i < 5; i++ )
        {
            StringTokenizer st = new StringTokenizer( in.readLine() );

            AB = Double.parseDouble( st.nextToken() );
            BC = Double.parseDouble( st.nextToken() );
            AC = Double.parseDouble( st.nextToken() );
            DE = Double.parseDouble( st.nextToken() );
            EF = Double.parseDouble( st.nextToken() );
            DF = Double.parseDouble( st.nextToken() );

            if( legal(AB, BC, AC) && legal(DE, EF, DF) )
                findProportionality();
            else
                System.out.println( "NOT VALID" );
        } // end for

        System.exit( 0 ); // don't omit this
    } // end main()

    private static boolean legal(double x, double y, double z)
    {
        double max = Math.max( Math.max( x,y ) , z ) ;

        if (max == x) if ( max < y + z ) return true ; // assumes max == x
        if (max == y) if ( max < x + z ) return true ; // assumes max == y
        if (max == z) if ( max < x + y ) return true ; // assumes max == z
        return false ;
    } // end legal()

    private static void findProportionality()
    {
        if ( AB/DE == BC/EF && AB/DE == AC/DF ) System.out.println("DEF");
        else if ( AB/DE == BC/DF && AB/DE == AC/EF ) System.out.println("EDF");

        else if ( AB/EF == BC/DE && AB/EF == AC/DF ) System.out.println("FED");
        else if ( AB/EF == BC/DF && AB/EF == AC/DE ) System.out.println("EFD");

        else if ( AB/DF == BC/DE && AB/DF == AC/EF ) System.out.println("FDE");
        else if ( AB/DF == BC/EF && AB/DF == AC/DE ) System.out.println("DFE");

        else System.out.println("NONE");
    }
} // // end class Triangles [ Senior Division ]

```