1. Determine the output of the following Java program.

```java
// Problem1.java
import java.util.Scanner;
import java.io.*;
class Problem1{
    public static void main(String[] args){
        int a=5, b=6, c=1;
        double x=0.5, y=1.0, z=1.5;

        c = fcn1(a, b);
        y = fcn2(y, a);
        b = fcn3(x, y);
        z = fcn3(c, b);
        System.out.println("a="+a+", b="+b+", c="+c);
        System.out.println("x="+x+", y="+y+", z="+z);
    }

    static int fcn1(int i, int j){
        int k = i-j;
        return (++k);
    }

    static double fcn2(double t, int n){
        return (t*n);
    }

    static int fcn3(double u, double v){
        return fcn1((int)(u*v), 2);
    }

    static double fcn3(int r, int s){
        return fcn2(r,s);
    }
}
```
2. Complete the static method `getMax()` in the following Java program in such a way that it returns the maximum value stored in its 2-dimensional array argument. Note that the array passed to `getMax()` need not be rectangular, i.e. its rows may have differing lengths.

```java
// Problem2.java
public static void main(String[] args){
    System.out.println(getMax(table)); // prints 54
}

static int getMax(int[][] A){
    // your code goes here
}
}
```

3. Determine the output of the following Java program. Assume that the program is run four times with the command line arguments `file1`, `file2`, `file3`, and `file4`, and assume that these files contain the following characters:

```
file1: zero 3 2 4 17 5 8 9 16 13 28 77 12
file2: one 4 3 72 0 9 2 17 5 19 50 8 91 14
file3: two 5 3 9 2 18 27 45 66 91 92 93
file4: blah 2 0 1 14 101 83 17 74 6 89 234
```

Print the program output only, do not print the Unix commands that invoke the program.

```java
// Problem3.java
import java.util.Scanner;
import java.io.*;
public static void main(String[] args) throws FileNotFoundException{
    Scanner sc = new Scanner(new File(args[0]));
    String str = sc.next();
    int n;
    while(sc.hasNextInt()){  
        n = sc.nextInt();
        if     ( str.equals("zero") && n%3==0 ) System.out.print(n + " ");  
        else if( str.equals("one") && n%3==1 ) System.out.print(n + " ");  
        else if( str.equals("two") && n%3==2 ) System.out.print(n + " ");  
        else continue;
    }
    System.out.println("end of " + str + "s");
}
```
4. Determine the output of the following Java program. (Hint: figure out what each part of the program does, particularly the methods Mix() and Print(), then put the pieces together.) Assume the program is invoked with one command line argument: % java Problem4 myFile, where myFile is a file containing the single line of text: one two three four five six seven eight nine ten

// Problem4.java
import java.util.Scanner;
import java.io.*;

class Problem4{
    public static void main(String[] args) throws FileNotFoundException {
        Scanner sc = new Scanner(new File(args[0]));
        String[] Z = new String[10];
        int[] A = {2, 1, 9, 8, 7, 6, 4, 5, 3, 0};
        
        for(int i=0; i<10; i++)  Z[i] = sc.next();
        Mix(Z, A);
        Print(Z);
    }

    static void Mix(String[] S, int[] I){
        int j, n = S.length;
        String[] temp = new String[n];
        
        for(j=0; j<n; j++) temp[j] = S[I[j]];
        for(j=0; j<n; j++) S[j] = temp[j];
    }

    static void Print(String[] S){
        for(int j=0; j<S.length; j++) System.out.println(S[j]);
    }
}
5. Determine what is printed by the following Java program, which consists of two files: Place.java and PlaceTest.java. Assume that both files are in the same directory.

// Place.java
class Place{
    // Fields
    String name; double latitude; double longitude;
    // Constructor
    Place(String n, double lat, double lon){
        name = n;
        latitude = lat;
        longitude = lon;
    }  
    public String toString(){
        return( name +": "+latitude+" N "+longitude+" W" );
    }
    boolean isFurtherWestThan(Place x){ return(this.longitude>x.longitude); }
    boolean isFurtherNorthThan(Place x){ return(this.latitude>x.latitude ); }
}

// PlaceTest.java
class PlaceTest{
    public static void main(String[] args){
        Place[] P = new Place[4];
        P[0] = new Place("Doriath", 33.1, 84.5);  
        P[1] = new Place("Nargothrond", 39.8, 101.7);
        P[2] = new Place("Thangorodrim", 45.0, 73.3);
        P[3] = new Place("Gondolin", 40.6, 74.2);
        int i, n;
        for(n=P.length; n>1; n--)
            for(i=0; i<n-1; i++)
                if( P[i].isFurtherNorthThan(P[i+1]) ) swap(P, i, i+1);
        printPlaceArray(P);
    }
    static void swap(Place[] A, int i, int j){
    }
    static void printPlaceArray(Place[] A){
        for(int i=0; i<A.length; i++) System.out.println(A[i]);
    }
}
6. The Java program below consists of two files: `Rational.java` and `RationalTest.java`, assumed to be in the same directory. The Rational class represents rational numbers as a pair of integers. Complete methods `plus()`, `times()` `toString()`, and `equals()` in `Rational.java` so that the output is:

7/5 + 13/3 = 86/15
7/5 * 13/3 = 91/15
7/5 * (13/3 + 28/20) = 2408/300
7/5 is not equal to 13/3
7/5 is equal to 28/20

// RationalTest.java
class RationalTest{
    public static void main(String[] args){
        String str;
        Rational x = new Rational(7, 5);
        Rational y = new Rational(13, 3);
        Rational z = new Rational(28, 20);
        System.out.println(x +"+"+ y +" = "+ x.plus(y) );
        System.out.println(x +"*"+ y +" = "+ x.times(y) );
        System.out.println(x +"*"+(y +"+"+ z +") = "+ x.times(y.plus(z)));
        str = (x.equals(y)?" is equal to ":" is not equal to ");
        System.out.println(x + str + y);
        str = (x.equals(z)?" is equal to ":" is not equal to ");
        System.out.println(x + str + z);
    }
}

// Rational.java
class Rational{
    int numerator; int denominator; // Fields
    Rational(int n, int d){ // Constructor
        if(d==0) throw new RuntimeException("zero denominator");
        numerator = n; denominator = d;
    }
    Rational plus(Rational Q){
    }
    Rational times(Rational Q){
    }
    public String toString(){
    }
    public boolean equals(Object x){
    }
} // end of class Rational
7. Complete the following java program which scans a file containing only numbers and white space, and prints out the average value of all the numbers in the file. For instance, given a file called `test` containing the numbers: 2.4 3.4 5.8 6.0 7 -3 12.2 8, and given that your program is called by doing `% java Problem7 test` at the command line, your program will produce the following output.

The average value in file test is 5.225

Your program will check that the number of command line arguments is equal to 1, and will initialize a `Scanner` object (in an appropriate try-catch block) pointing to the file named on the command line. If any of these tests fail, your program will exit with an error message. You may assume that the file will contain at least one number.

```java
// Problem7.java
import java.util.Scanner;
import java.io.*;
class Problem7{
    public static void main(String[] args){
        // your code begins here
        // your code ends here
    }
}
```
8. Determine the output of the following Java program consisting of the files: Itype.java, Atype.java, Btype.java and ItypeTest.java. Assume that all files are in the same directory.

// Itype.java
interface Itype{
    void foo();
}

// Atype.java
class Atype implements Itype{
    public void foo(){ System.out.println("in Atype's foo()"); }  
    public String toString(){ return "I am an Atype object"; }
}

// Btype.java
class Btype extends Atype{
    Btype(){ System.out.println("constructing a Btype object"); }  
    public void foo(){ System.out.println("in Btype's foo()"); }  
    public void bar(){ System.out.println("in Btype's bar()"); }  
    public String toString(){ return "I am a Btype object"; }
}

// ItypeTest.java
class ItypeTest{
    public static void main(String[] args){
        Itype I;
        Atype A;
        Btype B;

        A = new Atype();
        B = new Btype();
        I = A;
        A = B;
        I.foo();
        A.foo();
        B.bar();
        System.out.println(I);
        System.out.println(A);
        System.out.println(B);
    }
}