Problem 1

Write a program which reads four integers a, b, c and d; we assume that $a < b$, $c < d$ and all four numbers belong to closed interval $[-5, 5]$. For all integers n from this interval, the program prints information whether n belongs to the sum of closed intervals $[a, b]$ and $[c, d]$, to their intersection, and to their symmetric difference.

For example, for $a = -2, b = 3, c = 0, d = 4$, the result should be something like:

```
-5: inSum? false; inIntersect? false; inSymDiff? false
-4: inSum? false; inIntersect? false; inSymDiff? false
-3: inSum? false; inIntersect? false; inSymDiff? false
-2: inSum?  true; inIntersect? false; inSymDiff?  true
-1: inSum?  true; inIntersect? false; inSymDiff?  true
+0: inSum?  true; inIntersect?  true; inSymDiff? false
+1: inSum?  true; inIntersect?  true; inSymDiff? false
+2: inSum?  true; inIntersect?  true; inSymDiff? false
+3: inSum?  true; inIntersect?  true; inSymDiff? false
+4: inSum?  true; inIntersect? false; inSymDiff?  true
+5: inSum? false; inIntersect? false; inSymDiff? false
```

Do not use any **if** statements; work on boolean variables and values.

Problem 2

Write a program which reads one number of type **int** (say, n). Then it prints

1. Information whether the bit on the 7th position in the bit representation of n is set.
2. The number of bits set (which are 1) in the bit representation of n.
3. The position of the most significant bit which is set in n (or -1 if none is set).
4. The number which has the same bit representation as n, but with its two least significant bytes reversed.

Note: Most significant bits (bytes) are those which are written at the left and correspond to coefficients at the highest powers of 2. Least significant bits (bytes) are those which are written at the right and correspond to coefficients at the lowest powers of 2. The positions of bits are traditionally counted from 0 for the least significant bit (so they correspond to powers of 2).

Your program may have the following form:

download *BytesBits.java*

```java
import java.util.Scanner;

public class BytesBits {
    public static void main(String[] args) {
```

```
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter an int");
        int n = scan.nextInt();
        scan.close();
          // *1*
        boolean is7thBitSet = false;
              // your code here
        System.out.println("Is 7th bit set? " + is7thBitSet);
          // *2*
        int numberOf1s = 0;
              // your code here
        System.out.println("No. of 1s is " + numberOf1s);
          // *3*
        int mostSignificant = -1;
              // your code here
        System.out.println("Most significant bit set: " +
                           mostSignificant);
          // *4*
        int swapped = 0;
              // your code here
        System.out.println("With 2 least significant bytes" +
                " swapped: " + swapped);
    }
}
```

Examples of outputs:

```
Enter an int
1
Is 7th bit set? false
No. of 1s is 1
Most significant bit set: 0
With 2 least significant bytes swapped: 256

Enter an int
-1
Is 7th bit set? true
No. of 1s is 32
Most significant bit set: 31
With 2 least significant bytes swapped: -1

Enter an int
4080
Is 7th bit set? true
No. of 1s is 8
Most significant bit set: 11
```

```
With 2 least significant bytes swapped: 61455

Enter an int
511
Is 7th bit set? true
No. of 1s is 9
Most significant bit set: 8
With 2 least significant bytes swapped: 65281
```

Problem 3

A phone company stores information on one phone call in *one* 64-bit variable of type **long** which contains:

1. identifier of the calling customer (caller): 17-bit number, i.e., from interval $[0, 2^{17} - 1] = [0, 131071]$;
2. zone number of the caller (caller_zone): 7-bit number, i.e., from interval $[0, 2^7 - 1] = [0, 127]$;
3. identifier of the receiving customer (callee): 17-bit number;
4. zone number of the callee (callee_zone): 7-bit number;
5. duration of the call in seconds: 13-bit number, i.e., from interval $[0, 2^{13} - 1] = [0, 8191]$;
6. tariff number: 3-bit number, i.e., from interval $[0, 2^3 - 1] = [0, 7]$;

Write static functions

- **encode** — taking six numbers described above (as **int**s) and packing their values into *one* number of type **long**;
- **info** — printing information on one phone call passed to it as a single number of type **long**.

For example, the program

download *Telephones.java*

```java
public class Telephones {
    public static void main(String[] args) {
        info(encode(130999,101,7777,99,7000,6));
    }

    public static long encode(int caller, int caller_zone,
                              int callee, int callee_zone,
                              int duration, int tariff) {
        // ...
    }

    public static void info(long res) {
        // ...
    }
}
```

should print

```
Caller:      130999
Caller_zone: 101
Callee:      7777
Callee_zone: 99
Duration   : 7000
Tariff     : 6
```

Do *not* use any tools from packages other than ***java.lang***.
