Problem 1

Variable `a` (of type **int**) is declared and initialized with value 0xB7. Declare and initialize, using a literal in hexadecimal notation, variable `b` in such a way that printing the value of `a^b`, we get 513. Do not use loops.

Problem 2

Write a program which reads a natural odd number `n` greater than 1, and then prints something like the letter X with height `n`. For example, if the number is 7, the result should look like this:

```
*     *
 *   *
  * *
   *
  * *
 *   *
*     *
```

Problem 3

Let `arr` be the reference to an array of **int**s created in the program. We assume that its length is even.

The program reverses the order of the elements in the first half of the array and then prints it.

For example, if `arr` has elements $[1, 2, 3, 4, 5, 6]$, then after modification it should be $[3, 2, 1, 4, 5, 6]$.

After modifications of the size and/or values of elements of the array `arr`, the program should work correctly without any other changes.

Problem 4

Let `arr` be the reference to an array of **int**s created in the program.

The program checks if there are at least two pairs of consecutive elements with equal values, but for each pair this value is different.

For example, if thera are two pairs, say 7-7 and then again 7-7, this will not meet the requirement, but pairs 7-7 and 9-9 will.

After modifications of the size and/or values of elements of the array `arr`, the program should work correctly without any other changes.

Problem 5

Create a class containing static functions operating on arrays of type **int[]**:

- `static int[] add(int[] a, int elem)` — takes a sorted array `a` and creates and returns an array by one longer than `a` in which one element with value `elem` has been added. This new element should be added in such a way, that the resulting array is still sorted in nondecreasing order. Do not use sorting — the new element should be inserted into the resulting array directly on its correct location. The input array `a` may be of length 0.
- `static int[] delIndex(int[] a, int ind)` — takes an array `a` and creates and returns an array by one shorter than `a` in which there is no element which in `a` had index `ind`.
- `static int[] delFirst(int[] a, int e)` — takes an array `a` and creates and returns an array by one shorter than `a` in which there is no first element with value `e`. If there was no such element, the input array `a` is returned unmodified.
- `static int[] delLast(int[] a, int e)` — takes an array `a` and creates and returns an array by one shorter than `a` in which there is no the last element with value `e`. If there was no such element, the input array `a` is returned unmodified.
- `static int[] delAll(int[] a, int e)` — takes an array `a` and creates and returns a shorter array in which there is no elements with value `e`. If there was no such elements, the input array `a` is returned unmodified.
- `static void info(int[] a)` — takes an array `a` and prints it in one line with information about its size.

Remarks:

- Do not use any functions/classes from any packages except ***java.lang*** — in particular sorting or collections.
- Do not copy arrays or their subsequences using loops. Use instead the static method **arraycopy** from class **System** (from the ***java.lang*** package, no imports required)

  ```
  System.arraycopy(sArr, sIndex, tArr, tIndex, count)
  ```

  which copies `count` elements of the array `sArr` beginning at position `sIndex` to array `tArr` starting at position `tIndex`.
- In the **delIndex** function, the value of the index may be illegal — you don't have to handle such situation, just let the program crash.

For example, the following program

download *ArrAddRemove.java*

```
public class ArrAddRemove {
    public static void main(String[] args) {
        int[] a = {};
        info(a);          // line 1

        a = add(a, 4);
        a = add(a, 1);
        a = add(a, 3);
        a = add(a, 7);
```

```
        a = add(a, 4);
        a = add(a, 2);
        a = add(a, 7);
        a = add(a, 4);
        a = add(a, 8);
        a = add(a, 7);
        a = add(a, 4);
        a = add(a, 5);

        info(a);          // line 2

        a = delIndex(a, 2);
        a = delLast(a, 7);
        a = delFirst(a, 7);

        info(a);          // line 3

        a = delAll(a, 4);

        info(a);          // line 4
    }

    static int[] add(int[] a, int elem) {
        // ...
    }
    static int[] delIndex(int[] a, int ind) {
        // ...
    }
    static int[] delFirst(int[] a, int e) {
        // ...
    }
    static int[] delLast(int[] a, int e) {
        // ...
    }
    static int[] delAll(int[] a, int e) {
        // ...
    }
    static void info(int[] a) {
        // ...
    }
}
```

should print

```
Length 0: [ ]
Length 12: [ 1 2 3 4 4 4 4 5 7 7 7 8 ]
```

```
Length 9: [ 1 2 4 4 4 4 5 7 8 ]
Length 5: [ 1 2 5 7 8 ]
```

Problem 6

Write a static function **right** which takes a two-dimensional *rectangular* (by assumption) array of **int**s (we assume that the number of columns is even). The function creates and returns a new two-dimensional array which contains the "right part" of the original array, i.e., without the first half of columns. The function itself doesn't print anything!

For example, the following program

download *Arr2DRight.java*

```
import java.util.Arrays;
public class Arr2DRight {
    // ...
    public static void main (String[] args) {
        int[][] a = { {1,2,3,4,5,6},
                      {2,3,4,5,6,7},
                      {3,4,5,6,7,8},
                      {4,5,6,7,8,9} };
        for (int[] r : a)
            System.out.println(Arrays.toString(r));
        System.out.println();
        for (int[] r : right(a))
            System.out.println(Arrays.toString(r));
    }
}
```

should print

```
[1, 2, 3, 4, 5, 6]
[2, 3, 4, 5, 6, 7]
[3, 4, 5, 6, 7, 8]
[4, 5, 6, 7, 8, 9]

[4, 5, 6]
[5, 6, 7]
[6, 7, 8]
[7, 8, 9]
```

Problem 7

Write a function taking a two-dimensional, rectangular array of **int**s and zeroing all rows and columns which contain at least one zero element.

For example, the program

download *ZerosMatrix.java*

```java
public class ZerosMatrix {
    public static void main(String[] args){
        int[][] a = {
            {  4,  9, 10,  0,  1,  2 },
            {  7, -8, 20,  1,  5,  8 },
            {  1,  8,  3,  2,  1, -3 },
            {  1,  8, -3,  2, 11, -3 },
            { 17,  0,  5, -9, 21, 10 }
        };
        printArr(a, "Original matrix");
        setZeros(a);
        printArr(a, "Zeroing rows and columns containing zero");
    }

    public static void setZeros(int[][] arr) {
        // ...
    }
    public static void printArr(int[][] arr, String message) {
        // ...
    }
}
```

should print something like

```
Original matrix
      4  9 10  0  1  2
      7 -8 20  1  5  8
      1  8  3  2  1 -3
      1  8 -3  2 11 -3
     17  0  5 -9 21 10
Zeroing rows and columns containg zero
      0  0  0  0  0  0
      7  0 20  0  5  8
      1  0  3  0  1 -3
      1  0 -3  0 11 -3
      0  0  0  0  0  0
```

The function cannot create any two-dimensional arrays, but may create, no more than two, one-dimensional arrays.
