Problem One: Palindrome Finder

A palindrome is a string of characters which reads the same in both directions. For example, “abba” is a palindrome. Write a program that takes one line of input from the user and then find all palindromes in the string that are three or more characters. Your program should ignore white-spaces, punctuation and case. Your program should also output the character positions for the first and last character of each palindrome in the sequence. Here is an example run with the user input shown in boldface.

Example Run:

Please enter a string to analyze: The Madam who knows Weffelle more than I?

[4, 8] madam
[5, 7] ada
[17, 20] ws w
[21, 24] effe
[24, 27] elle

Analyze another string? (y/n) n

Notice that overlaps are allowed.

After analyzing a string, your program should ask if the user wants to analyze another string.
Problem Two: Caesar’s Cipher

Caesar’s Cipher is an encryption/decryption technique that is fairly simple but produces some results that are difficult to immediately decrypt. It works by shifting all the letters a set number of letters ahead in the alphabet. For instance, if we shifted by 5, then a becomes f, b becomes g, c becomes h, and so on. Using a shift of 5, the sentence “I am alive.” becomes “N FR FQNAJ.”

In order for us to be able to decrypt such a message we need to know what the shift was. We can add this in the message as a blank character and last letter. Using the above example, the encrypted message then becomes “N FR FQNAJ. E” as ‘E’ is the fifth letter in the alphabet.

Write a program that can both encrypt and decrypt strings. White-space and punctuation characters should not be encrypted.

When encrypting a string, the program should pick a random shift between 1 and 25.

When decrypting a string, the program should read the shift from the string itself (since the shift is encoded in the last letter).

Here is an example run with the user input shown in boldface. This run assumes that the shift that must be randomly chosen was 5.

Example Run:

Do you want to encrypt or decrypt? (or quit?) encrypt

Please enter a line of text to encrypt: I am alive.

Your encrypted message is: N FR FQNAJ. E

Do you want to encrypt or decrypt? (or quit?) decrypt

Please enter a line of text to decrypt: N FR FQNAJ. E

Your decrypted message is: I am alive.

Do you want to encrypt or decrypt? (or quit?) quit

Thank-you.
Problem Three: MinMax Sorting

You have discovered a new sorting algorithm! You named it MinMax Sorting. Your algorithm is passed an unsorted array of integers with indices 0 through \( n - 1 \). During the first pass you search sequentially all the array elements and find the index of the smallest value and the index of the largest value. If multiple indices have the smallest value (or the largest) it does not matter which one you use next. Use swaps so that the value in the index of the smallest value is swapped with the value in index 0. Swap the value in the index of the largest value with the value in index \( n - 1 \). You have to correctly handle the case when the smallest index or the largest index holds the smallest value or the largest value.

In the second pass repeat but just in the subarray with indices 1 through \( n - 2 \). In other words, at the end of the second pass you swap the smallest value you find in the second pass with the value in index 1 and you swap the largest value you find in the second pass with the value in index \( n - 2 \).

Continue passes until the subarray is one element or zero elements; the initial array could have an even or an odd number of elements.

Here is an example run with the user input shown in boldface. Your program output must show each pass in the format shown in this example.

Example Run:

Please enter the array of integers to be sorted: 89 45 68 90 29 34 17

89 45 68 90 29 34 17
Min Index: 6
Max Index: 3

17 45 68 89 29 34 90
Min Index: 4
Max Index: 3

17 29 68 34 45 89 90
Min Index: 3
Max Index: 2

The sorted array is:
17 29 34 45 68 89 90
Problem Four: Image Boundary Detection

Write a program that will support an image manipulation functionality with the image represented by an N-by-N array of double values. The class will have two fields each of which is two-dimensional array of double values. One field, called `image`, is the original array of values. The other field, called `newImage`, has an array which has the same number of rows and columns as the original array in which all the elements are either 0.0 or 1.0. Each 1.0 in the `newImage` array indicates that the corresponding value in the `image` array exceeds a threshold value times the average of all values in the `image` array. The other elements in the `newImage` array are 0.0.

For example, consider the following original array, `image`.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>1.3</td>
<td>4.5</td>
<td>6.0</td>
</tr>
<tr>
<td>1.7</td>
<td>3.3</td>
<td>4.4</td>
<td>10.5</td>
</tr>
<tr>
<td>1.1</td>
<td>4.5</td>
<td>2.1</td>
<td>25.3</td>
</tr>
<tr>
<td>1.0</td>
<td>9.5</td>
<td>8.3</td>
<td>2.9</td>
</tr>
</tbody>
</table>

The average value is 5.475. The resulting array, `newImage`, for a threshold of 1.4 would be

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

The average times the threshold is 5.475 x 1.4 which is 7.665. For example, consider element `image[0][3]`. Its value is 6.0 which is less than 7.665 so a 0.0 appears in the element `newImage[0][3].`

The resulting array, `newImage`, for a threshold of 0.6 would be

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
</tr>
<tr>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

The average times the threshold is 5.475 x 0.6 which is 3.285. For example, consider element `image[0][3]`. Its value is 6.0 which is greater than 3.285 so a 1.0 appears in the element `newImage[0][3].`

Your program must work with the main method listed below and which is also on the WebCat course account.

```java
public static void main(String[] args) {
    double[][] data = {{1.2, 1.3, 4.5, 6.0},
                       {1.7, 3.3, 4.4, 10.5},
                       {1.1, 4.5, 2.1, 25.3},
                       {1.0, 9.5, 8.3, 2.9}};

    ImageManip image = new ImageManip(data);
```
System.out.println("The original image array is "+image.originalImage());
image.findFigure(1.4);
System.out.println("The image array with a threshold of 1.4 is "+image.newImage());
image.findFigure(0.6);
System.out.println("The image array with a threshold of 0.6 is "+image.newImage());
}

An example run of the program is given below.
Your output should match mine as closely as possible for full credit.

The original image array is
1.2 1.3 4.5 6.0
1.7 3.3 4.4 10.5
1.1 4.5 2.1 25.3
1.0 9.5 8.3 2.9

The image array with a threshold of 1.4 is
0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0
0.0 0.0 0.0 1.0
0.0 1.0 1.0 0.0

The image array with a threshold of 0.6 is
0.0 0.0 1.0 1.0
0.0 1.0 1.0 1.0
0.0 1.0 0.0 1.0
0.0 1.0 1.0 0.0