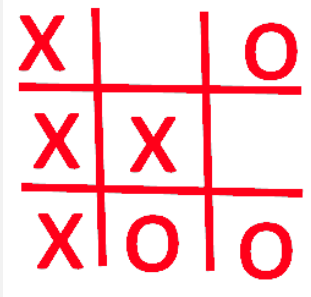
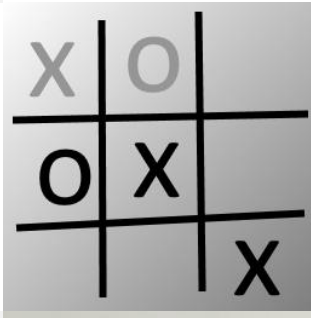
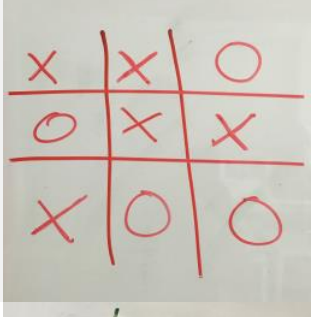
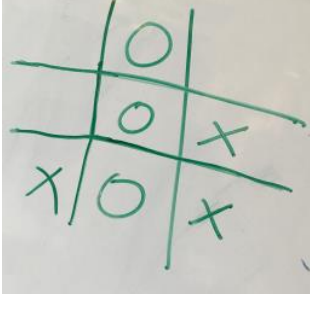


### Task 3: Tic-tac-toe [100 points]

Joe and Mike are playing [tic-tac-toe](#). They're quite old, their memory is failing them, so they need to record their games for future bragging rights. Joe likes to draw on a whiteboard and take a photo of the board, while Mike prefers to use MS Paint and save the image (so they sometimes use the whiteboard and other times they draw in MS Paint).

Your task is to write a program which would analyze the image (either whiteboard or MS Paint) and convert it to a more "digital" format (set of positions).

Some examples:

Image	Result
	X-O XX- XOO
	XO- OX- --X
	XXO OXX XOO
	-O- -OX XOX

Notes about the images:

1. All images are in BMP format (fixed size 256x256 pixels, 24 bits per pixel RGB)
2. Size of the board is always 3x3
3. Some images were drawn with the most basic MS Paint skills, using uniform color on a white background
4. Mike occasionally likes to touch up his MS Paint images, so the background might have a different shade of gray, maybe even some gradient. Objects could also be of slightly different dark shades.
5. Joe is old and can't hold the camera steady, so his photos of the whiteboard are occasionally rotated up to 30 degrees
6. Both Mike and Joe are very neat, the grid is a single connected component and objects don't "touch" each other; they're at least 5 pixels apart. The individual symbols are single connected components too.
7. Game board always takes up the majority of the image
8. MS Paint brushes, as well as whiteboard markers, are up to 10 pixels in width in the final image
9. Symbol bounding box will be at least 25x25 pixels.
10. There might be other (smaller) artifacts on the picture here and there (e.g. previous game's whiteboard marker strokes that Joe and Mike failed to erase completely). Anyway, their stroke width will still stay below 10 pixels, as for 'X' and 'O' symbols.

## Input

`inputImagePath outputTextPath` (provided as command line parameters).

## Output

Your program should write three lines of text into a text file, each containing three ASCII characters representing the board state from the input image. Each board position can be filled with either "X" or "O" or empty. These states are to be represented with ASCII characters 'X', 'O' and '-', respectively.

## Samples

In `Samples\TicTacToe\Input` folder, you will find several images along with their expected outputs in `Samples\TicTacToe\Output`

## Calling

In case you're producing Windows executable, your program will be called like:

```
ttt.exe inputImagePath outputTextPath
```

In case of Python script

```
Python ttt.py inputImagePath outputTextPath
```

In case of Java

```
Java -jar TTT.jar inputImagePath outputTextPath
```

or

```
java TTT inputImagePath outputTextPath
```

In case you wish to write an Octave function, it should be placed in a file named `ttt.m`, containing

```
function ttt (inputImagePath outputTextPath)
```

which performs the processing and outputs the result as described.

Time limit for task execution is 20 seconds per input image, on a standard 2GHz machine.

[Document update history](#)

5/20/2016 7:05 PM: Document created