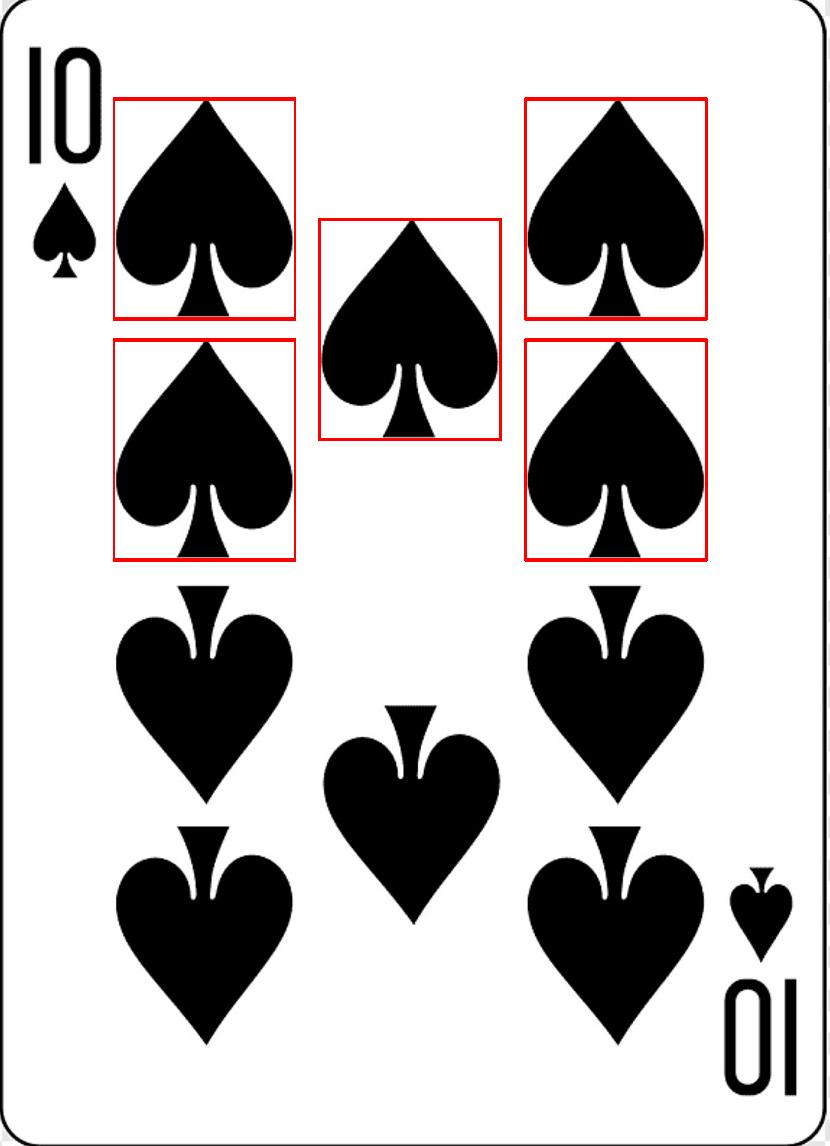
# Problem 1: Single Thread and Multi-Thread Implementation of Template Matching Algorithm

The resulting image from the basic algorithm:

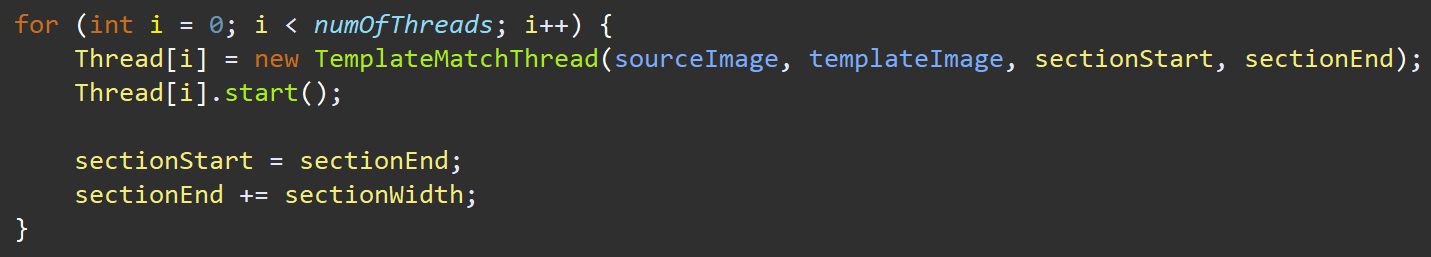
A screen shot of a computer program

Description automatically generatedWhere the algorithm is written similarly to the given Matlab-like pseudo-code.

Where the method convertImage converts from BufferedImage to short[][] to calculate the absolute difference.

## Multi-threaded implementation

By using a thread class, the algorithm can be done by having multiple concurrent threads scanning the source image for a match with the template image. The class is given by:  
public class TemplateMatchThread extends Thread{  
Where the Thread.start(); method is overridden to run the template matching algorithm.

This thread is invoked by the driver class, which creates a for loop and creates a new Thread object for every numOfThreads:

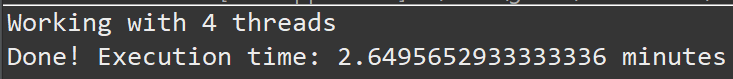
A new section of the image Is given to each thread, where it scans for a match between the template and the source.

Single-thread result:  
This is done by setting the numOfThreads = 1.  
A screen shot of a computer

Description automatically generated

Multi-threaded results:  
A screen shot of a number

Description automatically generated  
A black background with white text

Description automatically generated  


Note: With a higher number of threads, the template slice is unable to overlap with potential perfect matches before terminating; due to splitting even slices of the source image by the number of threads.

This information proves that using more threads can drastically cut the execution time down a lot. The higher number of threads, the lower the execution time. This is done in an inverse exponential curve.

The only issue with using more threads is the lesser likelihood of successful matches due to splitting the source image in sections.

Github repo: [gp974UOWD/gp974-ECTE331 (github.com)](https://github.com/gp974UOWD/gp974-ECTE331)