*** ***

***Automatic Bubble Sheet Exam Correction***

***Team Members:***

* ***Ahmed Hossny Hamed - Abd-Elrahman Ali Sayed***
* ***Mohamed Youssef Mostafa - Ahmed Alaa Ahmed***
* ***Ebrahim Ahmed Hamed - Mohamed Mostafa Ahmed***
* ***Sohair Ahmed Abd-Elraheen - Nermeen Abd-Elazeem Abd-Elrazek***

***Dr.Supervisor :***

***Dr: Khaled Fathi***

***Table of Contents***

Part 1

1.0 Introduction …………………………………………………………………..…………………………

1.1 Problem statement …………………………………………………….…………………………….

1.2 Methodology and steps …………………………………………..………………………………..

Part 2

2.0 What is a Computer vision ?.........................................................................

2.1 Computer vision Applications ..............................................................

Part 3

3.0 OpenCV library ……………….……………………………………………………………..……….

3.1 Grayscale image in openCV………………………….…………………………………

3.2.0 Gaussian blur in openCV ..................................................................

3.2.1 Sample Gaussian matrix ..........................................................

3.2.2 Common uses of Gaussian blur …………… ……………………………..

3.3.0Canny edge detector ………...............................................................

3.3.1 Process of Canny edge detection algorithm ….……………………..

3.4.0 Thresholding ………………………………………………………………………………..

3.4.1 What is thresholding ……………………………………………………………

3.4.2 Categorizing thresholding methods ……….…………………………..

Part 4

4.0 What is NumPy? ...........................................................................................

4.1 Why Use NumPy ? …………………..….…………………………………………………..

4.2 Why is NumPy Faster Than Lists……………………………………………………….

Part 5

5.0 Optical mark recognition …….......................................................................

5.1 Optical answer sheet …..…………………………………………………………………..

5.2 OMR software ……………………….……………………………………………………….

Part 6

6.0 Tool identification …………………………………………………………………………………….

Part 7

7.0 Tool identification …………………………………………………………………………………….

Part 8

8.0 References ……………………………………………..……………………………………………….

1. ***Introduction***

We explore an image-processing algorithm for auto grading of answer sheets using the mobile phone. Multiple-choice questions are widespread mechanism used by schools and universities to test student performance. They are common on standardized tests. Big organizations use OMR forms coupled with OMR software and dedicated scanner for grading multiplechoice questions. But small institutes and individual teachers cannot afford this costly setup and have to do time consuming manual grading. So we propose a mobile application that can be used by anyone at any time to grade answer sheets and save the valuable time spent grading manually. First we will discuss the OpenCV implementation of the image-processing algorithm and report on its challenges. Second we will give an overview of what was implemented on the android phone. Finally we will summarize the challenges and possibilities for future work.

* 1. ***Problem Statement***

Throughout the world, multiple-choice questions have become an integral part of the educational system. Standardized tests also use multiplechoice questions to judge student's academic performance. The American College Testing (ACT), the Scholastic Aptitude Test (SAT) and Law School Admission Test (LSAT) are just some of the many standardized tests conducted in the US to assess candidates on a common platform. Every year millions of students take standardized tests and they have to answer various questions asked by darkening bubbles in OMR sheets. Current solutions for evaluating these OMR sheets are expensive and need dedicated scanner, OMR software and buying customized OMR sheets.

So small organizations, institutes, individual teachers and tutors cannot use this convenient method of grading without spending lot of money. They resort to manually grading answer sheets. To grade a standardized test responses of a student takes 10 minutes on an average. The motivation of our project comes from the idea that we could build a mobile app that will assist the instructors in auto grading these answer sheets and saving their valuable instruction time. Now a day’s smartphones are ubiquitous. So there will be no extra cost associated with using this smartphone-based solution.

* 1. ***Methodology and steps***

**The steps to build a bubble sheet scanner and grader:**

The goal of this blog post is to build a bubble sheet scanner and test grader :

To accomplish this, our implementation will need to satisfy the following 7 steps:

* **Step #1:** Detect the exam in an image.
* **Step #2:** Apply a perspective transform to extract the top-down, birds-eye-view of the exam.
* **Step #3:** Extract the set of bubbles (i.e., the possible answer choices) from the perspective transformed exam.
* **Step #4:** Sort the questions/bubbles into rows.
* **Step #5:** Determine the marked (i.e., “bubbled in”) answer for each row.
* **Step #6:** Lookup the correct answer in our answer key to determine if the user was correct in their choice.
* **Step #7:** Repeat for all questions in the exam.

***2.0 What is a Computer vision?***

Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do. "Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding." As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems.[1]

***2.1 Computer vision Applications :***

Applications range from tasks such as industrial machine vision systems which, say, inspect bottles speeding by on a production line, to research into artificial intelligence and computers or robots that can comprehend the world around them. The computer vision and machine vision fields have significant overlap. Computer vision covers the core technology of automated image analysis which is used in many fields. Machine vision usually refers to a process of combining automated image analysis with other methods and technologies to provide automated inspection and robot guidance in industrial applications.

In many computer-vision applications, the computers are pre-programmed to solve a particular task, but methods based on learning are now becoming increasingly common. Examples of applications of computer vision include systems for:

Learning 3D shapes has been a challenging task in computer vision. Recent advances in deep learning has enabled researchers to build models that are able to generate and reconstruct 3D shapes from single or multi-view depth maps or silhouettes seamlessly and efficiently [2]

* Automatic inspection, e.g., in manufacturing applications.
* Assisting humans in identification tasks, e.g., a species identification system.
* Controlling processes, e.g., an industrial robot.
* Detecting events, e.g., for visual surveillance or people counting, e.g., in the restaurant industry.
* Interaction, e.g., as the input to a device for computer-human interaction.
* Modeling objects or environments, e.g., medical image analysis or topographical modeling.
* Navigation, e.g., by an autonomous vehicle or mobile robot.
* Organizing information, e.g., for indexing databases of images and image sequences.

***3.0 OpenCV library:***

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. The library is used extensively in companies, research groups and by governmental bodies.

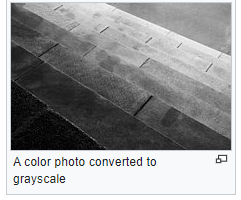
It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, [Android](https://opencv.org/android/) and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured [CUDA](https://opencv.org/cuda/)and [OpenCL](https://opencv.org/opencl/) interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms.

OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.[3]

***3.1 Grayscale image in openCV:***

A grayscale (or graylevel) image is simply one in which the only colors are shades of gray. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel. In fact a `gray' color is one in which the red, green and blue components all have equal intensity in RGB space, and so it is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full color image.

A grayscale image is one in which the value of each pixel is a single sample representing only an amount of light; that is, it carries only intensity information. Grayscale images, a kind of black-and-white or gray monochrome, are composed exclusively of shades of gray. The contrast ranges from black at the weakest intensity to white at the strongest.[4]

***3.2 Gaussian blur in openCV:***

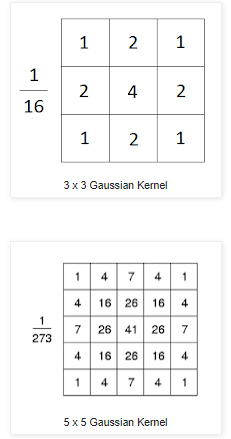
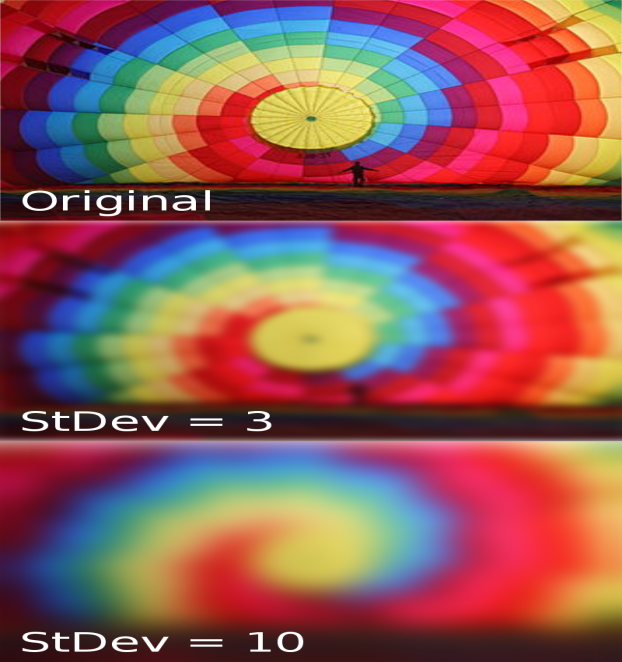
A **Gaussian blur** (also known as **Gaussian smoothing**) is the result of blurring an image by a [Gaussian function](https://en.wikipedia.org/wiki/Gaussian_function) (named after mathematician and scientist [Carl Friedrich Gauss](https://en.wikipedia.org/wiki/Carl_Friedrich_Gauss)). It is a widely used effect in graphics software, typically to reduce [image noise](https://en.wikipedia.org/wiki/Image_noise) and reduce detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the [image](https://en.wikipedia.org/wiki/Image) through a translucent screen, distinctly different from the [bokeh](https://en.wikipedia.org/wiki/Bokeh" \o "Bokeh) effect produced by an out-of-focus lens or the shadow of an object under usual illumination. Gaussian smoothing is also used as a pre-processing stage in [computer vision](https://en.wikipedia.org/wiki/Computer_vision) algorithms in order to enhance image structures at different scales—see [scale space representation](https://en.wikipedia.org/wiki/Scale_space_representation) and [scale space implementation](https://en.wikipedia.org/wiki/Scale_space_implementation).

Gaussian blur/smoothing is the most commonly used smoothing technique to eliminate noises in images and videos. In this technique, an image should be convolved with a Gaussian kernel to produce the smoothed image.  
  
 You may define the size of the kernel according to your requirement. But the standard deviation of the Gaussian distribution in X and Y direction should be chosen carefully considering the size of the kernel such that the edges of the kernel is close to zero. Here I have shown the 3 x 3 and 5 x 5 Gaussian kernels.

The Gaussian blur is a type of image-blurring filters that uses a Gaussian function (which also expresses the [normal distribution](https://en.wikipedia.org/wiki/Normal_distribution) in statistics) for calculating the [transformation](https://en.wikipedia.org/wiki/Transformation_(mathematics)) to apply to each [pixel](https://en.wikipedia.org/wiki/Pixel) in the image. The formula of a Gaussian function in one dimension is 

In two dimensions, it is the product of two such Gaussian functions, one in each dimension: 

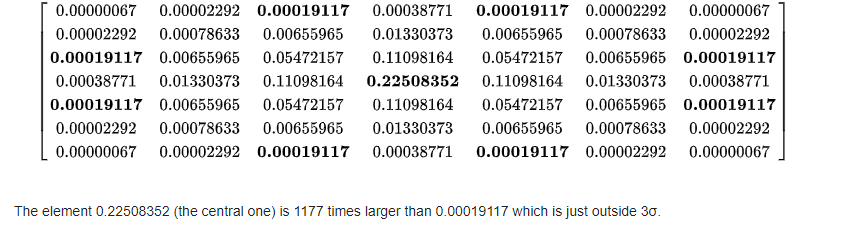
where *x* is the distance from the origin in the horizontal axis, *y* is the distance from the origin in the vertical axis, and *σ* is the [standard deviation](https://en.wikipedia.org/wiki/Standard_deviation) of the Gaussian distribution.[5]

***3.2.1 Sample Gaussian matrix:***

This sample matrix is produced by sampling the Gaussian filter kernel (with

σ = 0.84089642) at the midpoints of each pixel and then normalizing. The center element (at [4, 4]) has the largest value, decreasing symmetrically as distance from the center increases.[6]



***3.2.2*** ***Common uses of*** ***Gaussian blur***

This shows how smoothing affects edge detection. With more smoothing, fewer edges are detected

Gaussian smoothing is commonly used with [edge detection](https://en.wikipedia.org/wiki/Edge_detection). Most edge-detection algorithms are sensitive to noise; the 2-D Laplacian filter, built from a discretization of the [Laplace operator](https://en.wikipedia.org/wiki/Laplace_operator), is highly sensitive to noisy environments. Using a Gaussian Blur filter before edge detection aims to reduce the level of noise in the image, which improves the result of the following edge-detection algorithm. This approach is commonly referred to as [Laplacian of Gaussian](https://en.wikipedia.org/wiki/Laplacian_of_Gaussian), or LoG filtering

You can perform this operation on an image using the **Gaussianblur()** method of the **imgproc** class. Following is the syntax of this method [7]

GaussianBlur(src, dst, ksize, sigmaX)

***3.3.0 Canny edge detector:***

The **Canny edge detector** is an [edge detection](https://en.wikipedia.org/wiki/Edge_detection) operator that uses a multi-stage [algorithm](https://en.wikipedia.org/wiki/Algorithm) to detect a wide range of edges in images. It was developed by [John F. Canny](https://en.wikipedia.org/wiki/John_F._Canny) in 1986. Canny also produced a *computational theory of edge detection* explaining why the technique works.

Canny edge detection is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed. It has been widely applied in various [computer vision](https://en.wikipedia.org/wiki/Computer_vision) systems. Canny has found that the requirements for the application of [edge detection](https://en.wikipedia.org/wiki/Edge_detection) on diverse vision systems are relatively similar.

Thus, an edge detection solution to address these requirements can be implemented in a wide range of situations. The general criteria for edge detection include:

1. Detection of edge with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible
2. The edge point detected from the operator should accurately localize on the center of the edge.
3. A given edge in the image should only be marked once, and where possible, image noise should not create false edges.

To satisfy these requirements Canny used the [calculus of variations](https://en.wikipedia.org/wiki/Calculus_of_variations) – a technique which finds the [function](https://en.wikipedia.org/wiki/Function_(mathematics)) which optimizes a given [functional](https://en.wikipedia.org/wiki/Functional_(mathematics)). The optimal function in Canny's detector is described by the sum of four [exponential](https://en.wikipedia.org/wiki/Exponential_function) terms, but it can be approximated by the first [derivative](https://en.wikipedia.org/wiki/Derivative) of a [Gaussian](https://en.wikipedia.org/wiki/Gaussian_function).

Among the edge detection methods developed so far, Canny edge detection algorithm is one of the most strictly defined methods that provides good and reliable detection. Owing to its optimality to meet with the three criteria for edge detection and the simplicity of process for implementation, it became one of the most popular algorithms for edge detection. [8]

***3.3.1 Process of Canny edge detection algorithm:***

The Process of Canny edge detection algorithm can be broken down to 5 different steps:

1. Apply [Gaussian filter](https://en.wikipedia.org/wiki/Gaussian_filter) to smooth the image in order to remove the noise.
2. Find the intensity gradients of the image.
3. Apply non-maximum suppression to get rid of spurious response to edge detection.
4. Apply double threshold to determine potential edges.
5. Track edge by [hysteresis](https://en.wikipedia.org/wiki/Hysteresis): Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.[8]

***3.4.0 Thresholding:***

In [digital image processing](https://en.wikipedia.org/wiki/Digital_image_processing), **thresholding** is the simplest method of [segmenting images](https://en.wikipedia.org/wiki/Image_segmentation). From a [grayscale](https://en.wikipedia.org/wiki/Grayscale) image, thresholding can be used to create [binary images](https://en.wikipedia.org/wiki/Binary_image).

[](https://en.wikipedia.org/wiki/File:Pavlovsk_Railing_of_bridge_Yellow_palace_Winter.jpg) [](https://en.wikipedia.org/wiki/File:Pavlovsk_Railing_of_bridge_Yellow_palace_Winter_bw_threshold.jpg)

Original image Example of a threshold effect used on an image

***3.4.1 What is thresholding:***

The simplest thresholding methods replace each pixel in an image with a black pixel if the image intensity is less than some fixed constant T (that is, {\displaystyle I\_{i,j}<T}<T), or a white pixel if the image intensity is greater than that constant. In the example image on the right, this results in the dark tree becoming completely black, and the white snow becoming completely white.

***3.4.2 Categorizing thresholding methods:***

To make thresholding completely automated, it is necessary for the computer to automatically select the threshold T. categorize thresholding methods into the following six groups based on the information the algorithm manipulates .[9]

* [**Histogram**](https://en.wikipedia.org/wiki/Histogram)**shape**-based methods, where, for example, the peaks, valleys and curvatures of the smoothed histogram are analyzed.
* **Clustering**-based methods, where the gray-level samples are clustered in two parts as background and foreground (object), or alternately are modeled as a mixture of two Gaussians.
* [**Entropy**](https://en.wikipedia.org/wiki/Entropy_(information_theory))-based methods result in algorithms that use the entropy of the foreground and background regions, the cross-entropy between the original and binarized image, etc.
* **Object Attribute**-based methods search a measure of similarity between the gray-level and the binarized images, such as fuzzy shape similarity, edge coincidence, etc.
* **Spatial** methods [that] use higher-order probability distribution and/or correlation between pixels.
* **Local** methods adapt the threshold value on each pixel to the local image characteristics. In these methods, a different T is selected for each pixel in the image.[9]

***4.0 What is NumPy?***

NumPy is a python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy stands for Numerical Python.[10]

***4.1 Why Use NumPy ?***

In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is up to 50x faster that traditional Python lists.

The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.[10]

***4.2 Why is NumPy Faster Than Lists?***

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.This behavior is called locality of reference in computer science.

This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.[10]

***5.0 Optical mark recognition :***

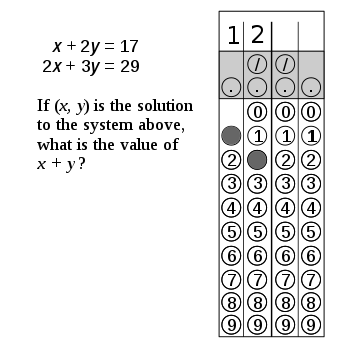
**Optical mark recognition** (also called **optical mark reading** and **OMR**) is the process of capturing human-marked data from [document forms](https://en.wikipedia.org/wiki/Form_(document)) such as surveys and tests. They are used to read questionnaires, multiple choice examination paper in the form of lines or shaded areas.

Many traditional OMR devices work with a dedicated [scanner](https://en.wikipedia.org/wiki/Image_scanner) device that shines a beam of light onto the form paper. The contrasting [reflectivity](https://en.wikipedia.org/wiki/Reflectivity) at predetermined positions on a page is then used to detect these marked areas because they reflect less light than the blank areas of the paper.

Some OMR devices use forms that are preprinted onto "transoptic" paper and measure the amount of light which passes through the paper; thus a mark on either side of the paper will reduce the amount of light passing through the paper.

***5.1 Optical answer sheet:***

An **optical answer sheet** or **bubble sheet** is a special type of [form](https://en.wikipedia.org/wiki/Form_(document)) used in [multiple choice question](https://en.wikipedia.org/wiki/Multiple_choice_question) [examinations](https://en.wikipedia.org/wiki/Test_(assessment)). Optical mark recognition is used to detect answers. The most well known company in the United States involved with optical answer sheets is the [Scantron Corporation](https://en.wikipedia.org/wiki/Scantron_Corporation" \o "Scantron Corporation), although certain uses require their own customized system.

[](https://en.wikipedia.org/wiki/File:SAT-Grid-In-Example.svg)

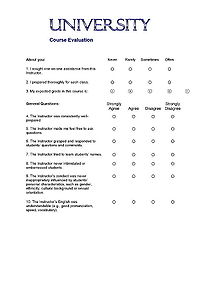
A response to an [SAT](https://en.wikipedia.org/wiki/SAT) math question marked on an optical answer sheet

Optical answer sheets usually have a set of blank ovals or boxes that correspond to each question, often on separate sheets of paper. [Bar codes](https://en.wikipedia.org/wiki/Bar_code) may mark the sheet for automatic processing, and each series of ovals filled will return a certain value when read. In this way students' answers can be digitally recorded, or identity given.

***5.2 OMR software:***

OMR software is a computer software application that makes OMR possible on a desktop computer by using an [Image scanner](https://en.wikipedia.org/wiki/Image_scanner) to process surveys, tests, attendance sheets, checklists, and other plain-paper forms printed on a laser printer.

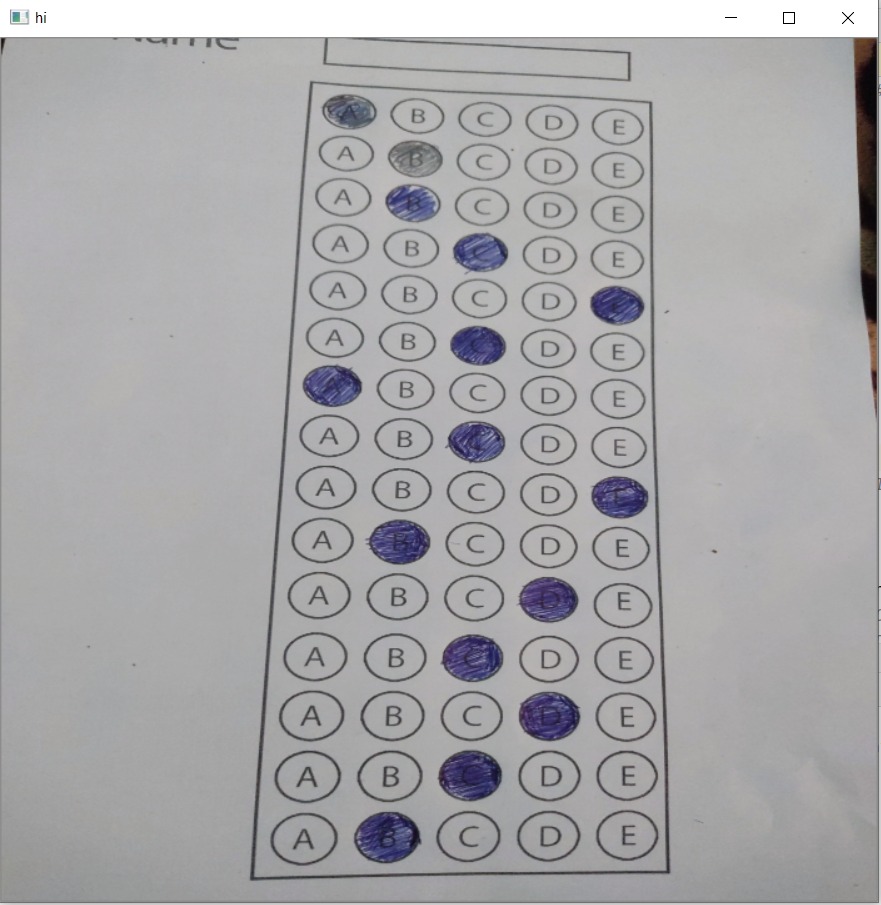
OMR software is used to capture data from OMR sheets. While data capturing scanning devices focus on many factors like thickness of paper dimensions of OMR sheet and designing pattern

[](https://en.wikipedia.org/wiki/File:PlainPaperOMRFormSm.jpg)

Plain paper OMR survey form, without registration marks and drop-out colors, designed to be scanned by an image scanner and OMR software

***6.0 How our modle work:***

- We need to get bubble sheet test image that we are going to grade for correctness so that we load our image from gallery or taken by camera.



- then defines our ANSWER\_KEY, the ANSWER\_KEY provides integer mappings of the question numbers to the index of the correct bubble.

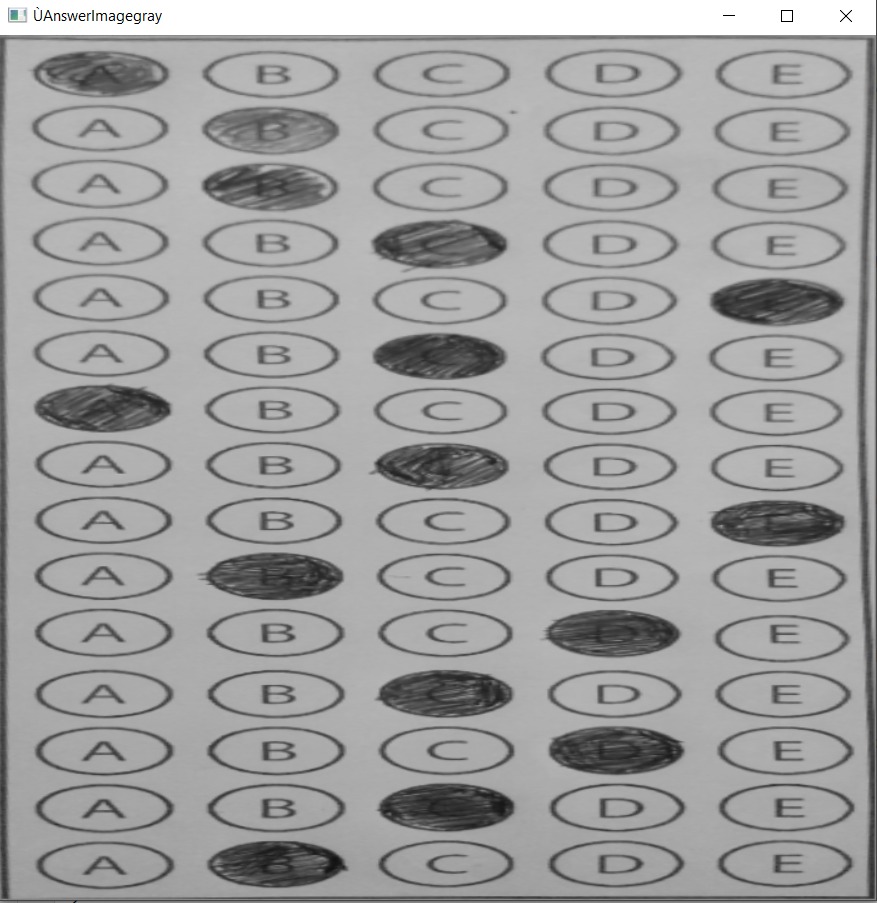
* Then the image sent to the server to preprocess.

ON SERVER :

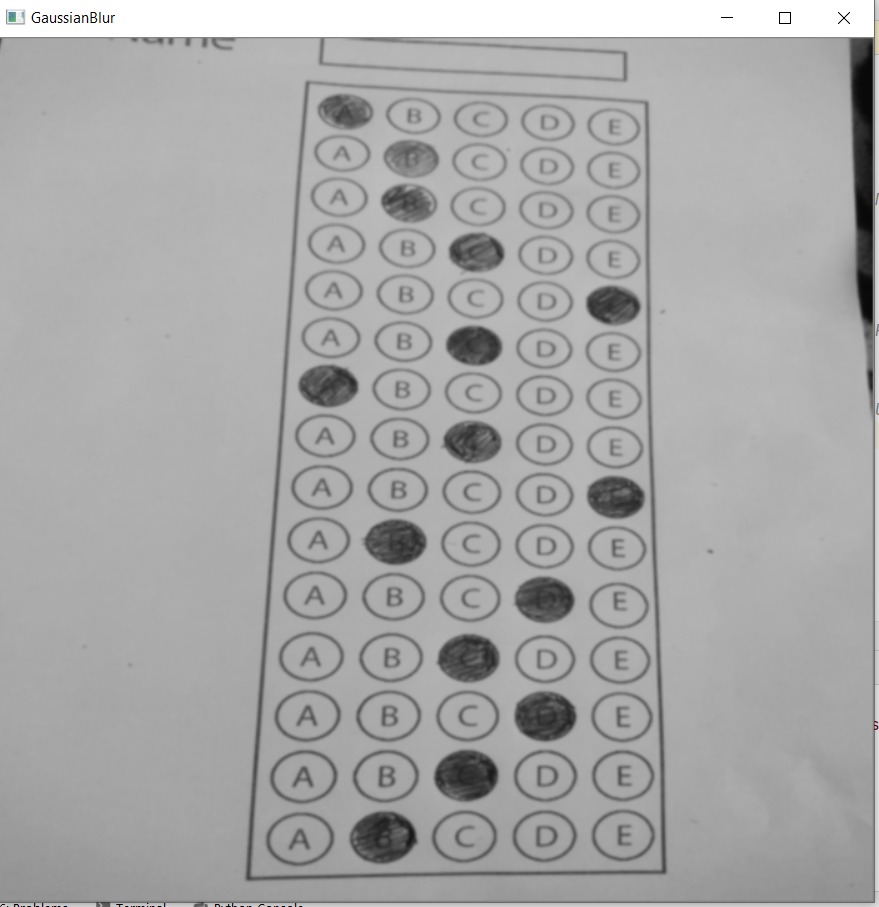
* First compress the image to reduse the size by

img = cv2.resize(img, (widthImg, heightImg))

* followed by converting it to grayscale ,and blurring it to reduce high frequency noise.
* imgGray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY) # CONVERT IMAGE TO GRAY SCALE



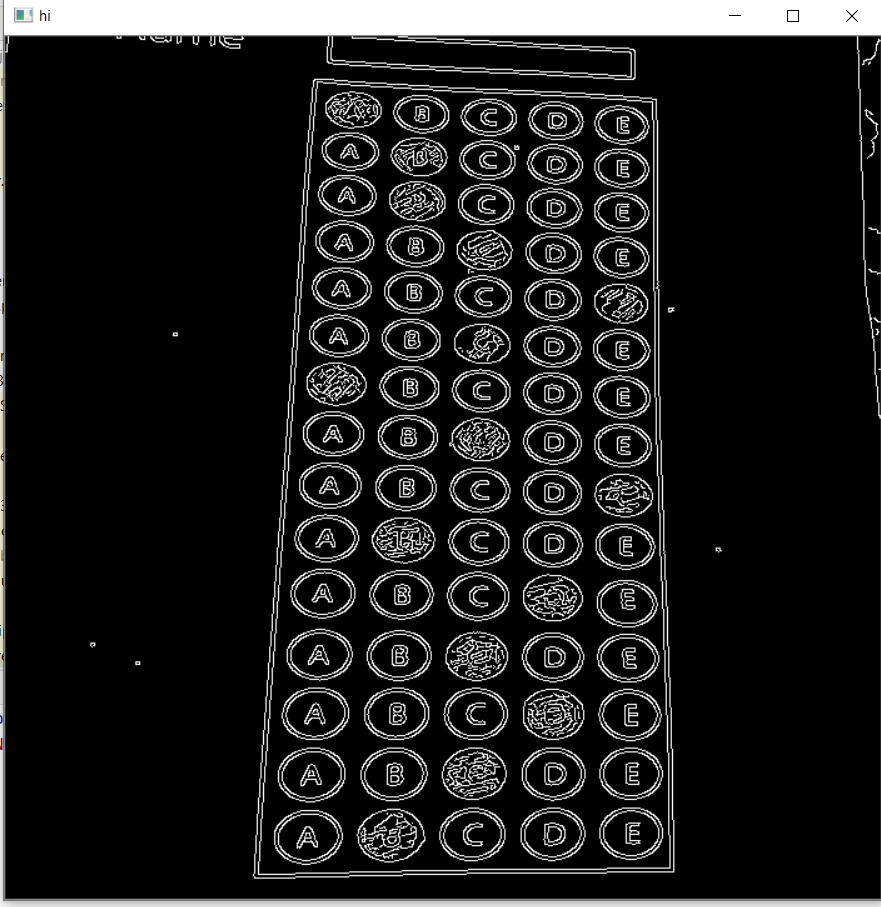
- imgBlur = cv2.GaussianBlur(imgGray, (5, 5), 1) # ADD GAUSSIAN BLUR



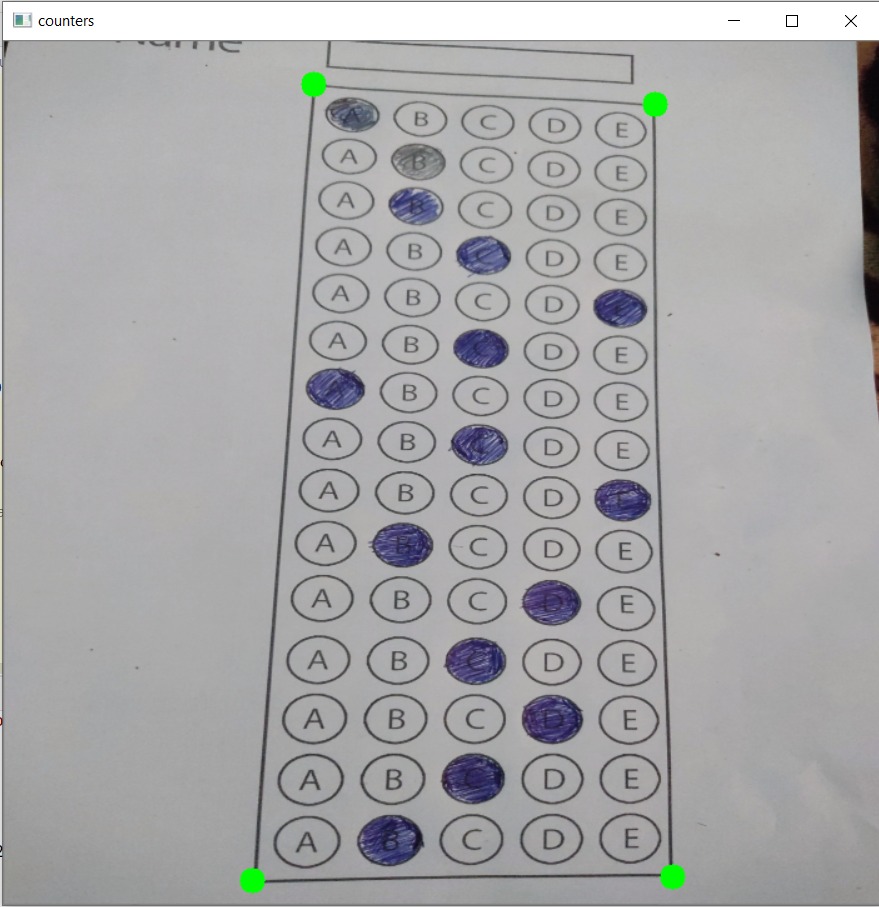
* We then apply the Canny edge detector to find the *edges/outlines* of the exam.

imgCanny = cv2.Canny(imgBlur, 10, 70) # APPLY CANNY

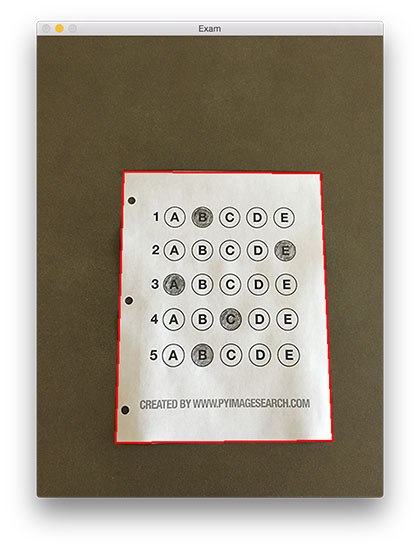
Below I have included a screenshot of our exam after applying edge detection:



* Now that we have the outline of our exam, we apply the cv2.findContours function to find the lines that correspond to the exam itself.



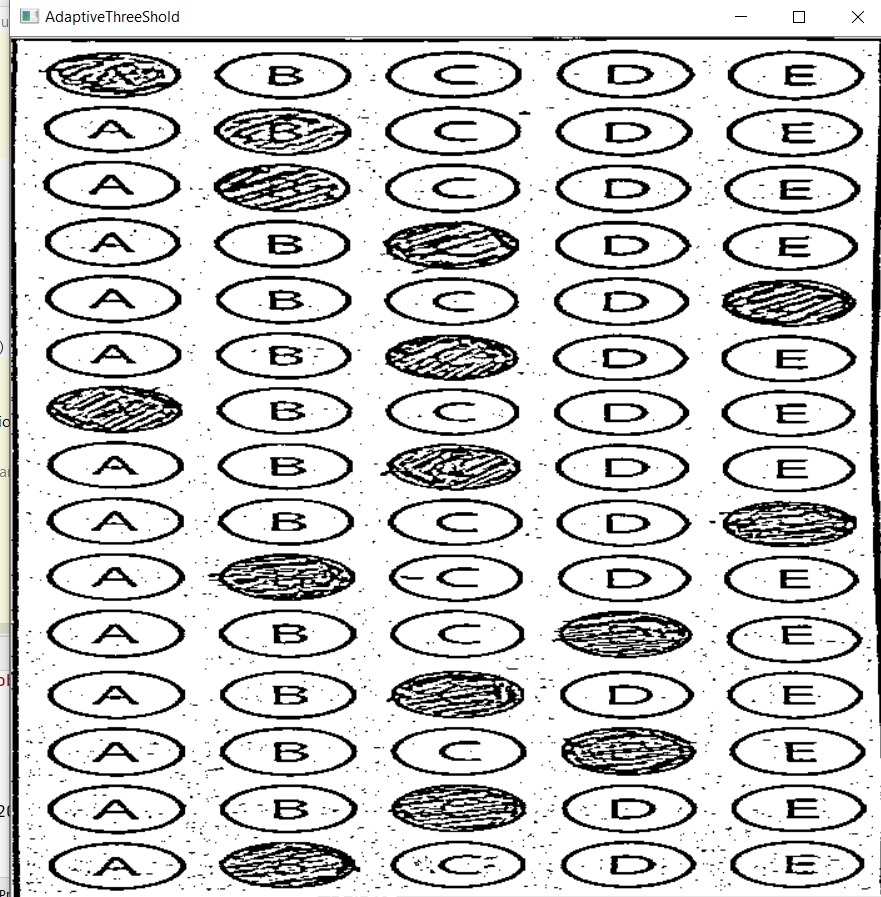
* We do this by sorting our contours by their area (from largest to smallest). This implies that larger contours will be placed at the front of the list, while smaller contours will appear farther back in the list.
* We make the assumption that our exam will be the main focal point of the image, and thus be larger than other objects in the image. This assumption allows us to “filter” our contours, simply by investigating their area and knowing that the contour that corresponds to the exam should be near the front of the list.
* However, contour area and size is not enough — we should also check the number of vertices on the contour.
* we make a check to see if our approximated contour has four points, and if it does, we assume that we have found the exam.

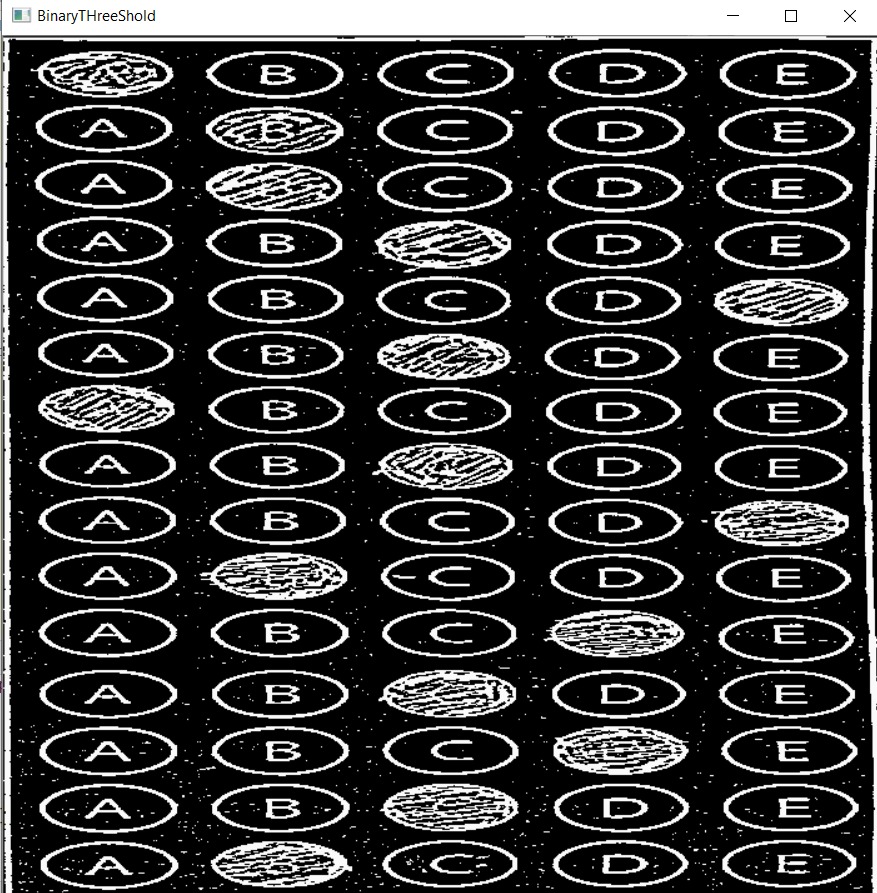


* Now that we have used contours to find the outline of the exam, we can apply a perspective transform to obtain a top-down, birds-eye-view of the document:
* Applying process of thresholding/segmenting the *foreground* from the *background* of the image:

imgThresh = cv2.threshold(imgWarpGray, 170, 255, cv2.THRESH\_BINARY\_INV)

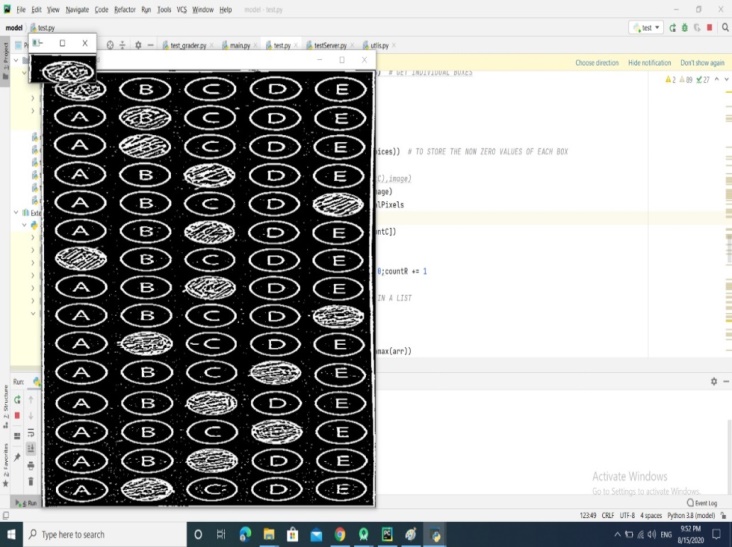
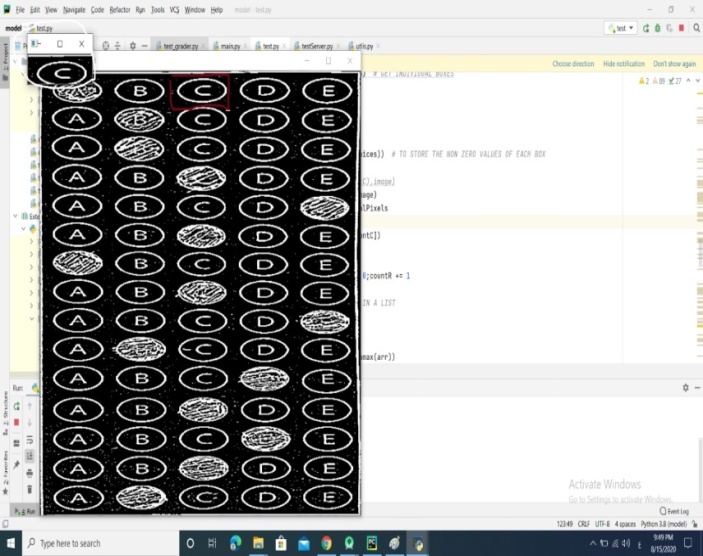
After applying Otsu’s thresholding method, our exam is now a *binary* image:

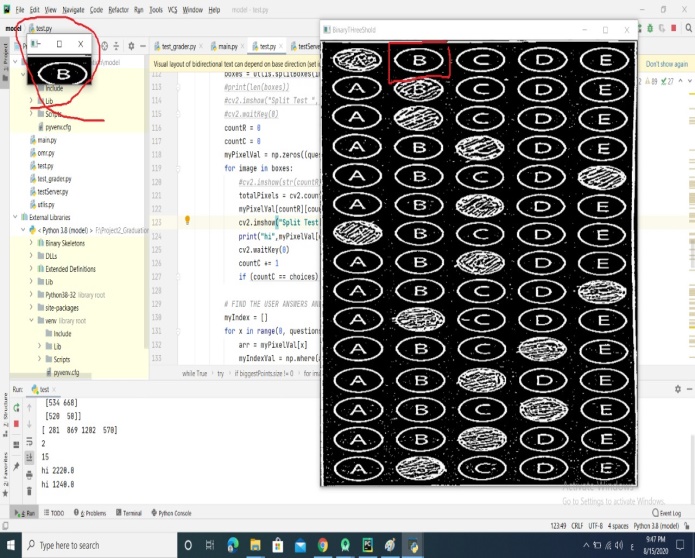


****

* For each of these pixels, we compute the bounding box ( which also allows us to compute the *aspect ratio*, or more simply, the ratio of the width to the height .
* Then FIND THE USER ANSWERS AND PUT THEM IN A LIST

We can now move on to the “grading” portion of our OMR system:



Based on whether the test taker was correct or incorrect yields which color is drawn on the exam. If the test taker is correct, we’ll highlight their answer in green. However, if the test taker made a mistake and marked an incorrect answer, we’ll let them know by highlighting the correct answer in red:

Finally, our last code block handles scoring the exam and sending to our android application and displaying the results to our screen:

***7.0 Tools identification***

(1) Python programming language

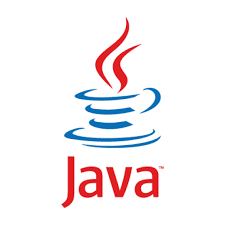
(2) server socket

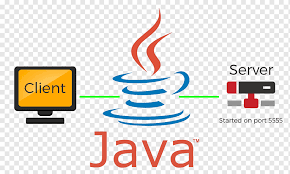
(3) Spyder

(4) Java programming languages

(5) Android studio

(6) Net Beans

***8.0 References***

[1] Dana H. Ballard; Christopher M. Brown (1982). Computer Vision. Prentice Hall. ISBN 978-0-13-165316-0.

[2]  *Wäldchen, Jana; Mäder, Patrick (2017-01-07).*[*"Plant Species Identification Using Computer Vision Techniques: A Systematic Literature Review"*](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6003396)*. Archives of Computational Methods in Engineering.****25****(2): 507–543.*[*doi*](https://en.wikipedia.org/wiki/Doi_(identifier))*:*[*10.1007/s11831-016-9206-z*](https://doi.org/10.1007%2Fs11831-016-9206-z)*.*[*ISSN*](https://en.wikipedia.org/wiki/ISSN_(identifier))[*1134-3060*](https://www.worldcat.org/issn/1134-3060)*.*[*PMC*](https://en.wikipedia.org/wiki/PMC_(identifier))[*6003396*](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6003396)*.*[*PMID*](https://en.wikipedia.org/wiki/PMID_(identifier))[*29962832*](https://pubmed.ncbi.nlm.nih.gov/29962832)*.*

[3] <https://opencv.org/about/>

[4]  *Johnson, Stephen (2006).*[*Stephen Johnson on Digital Photography*](https://books.google.com/books?id=0UVRXzF91gcC&pg=PA17&dq=grayscale+black-and-white-continuous-tone&ei=XlwqSdGVOILmkwTalPiIDw)*. O'Reilly.*[*ISBN*](https://en.wikipedia.org/wiki/ISBN_(identifier))[*0-596-52370-X*](https://en.wikipedia.org/wiki/Special:BookSources/0-596-52370-X)*.*

[5]  [Shapiro, L. G.](https://en.wikipedia.org/wiki/Linda_Shapiro" \o "Linda Shapiro) & Stockman, G. C: "Computer Vision", page 137, 150. Prentice Hall, 2001

[6] <https://en.wikipedia.org/wiki/Gaussian_blur#cite_note-6>

[7 ]Fisher, Perkins, Walker & Wolfart (2003). ["Spatial Filters - Laplacian of Gaussian"](http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm). Retrieved 2010-09-13

[8] Canny, J., [*A Computational Approach To Edge Detection*](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.420.3300&rep=rep1&type=pdf), IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

[9]  *Zhang, Y. (2011).*[*"Optimal multi-level Thresholding based on Maximum Tsallis Entropy via an Artificial Bee Colony Approach"*](https://doi.org/10.3390/e13040841)*. Entropy.****13****(4): 841–859.*[*Bibcode*](https://en.wikipedia.org/wiki/Bibcode_(identifier))*:[2011Entrp..13..841Z](https://ui.adsabs.harvard.edu/abs/2011Entrp..13..841Z).*[*doi*](https://en.wikipedia.org/wiki/Doi_(identifier))*:*[*10.3390/e13040841*](https://doi.org/10.3390%2Fe13040841)*.*

[10]<https://www.w3schools.com/python/numpy_intro.asp#:~:text=NumPy%20is%20a%20python%20library,NumPy%20stands%20for%20Numerical%20Python.>