מטלה1: מגישים: אופק כהן 319001533, לירן ענתבי 207435009

Problem1:

א. פיזי: יותר מגה פיקסלים מספקים איכות תמונה טובה יותר (יותר פרטים נראים לעין) אז זה השיקול העיקרי בפן הזה, לשם מה התמונה משמשת והאם חשוב שהפרטים הקטנים יהיו מאוד ברורים בה או לא.

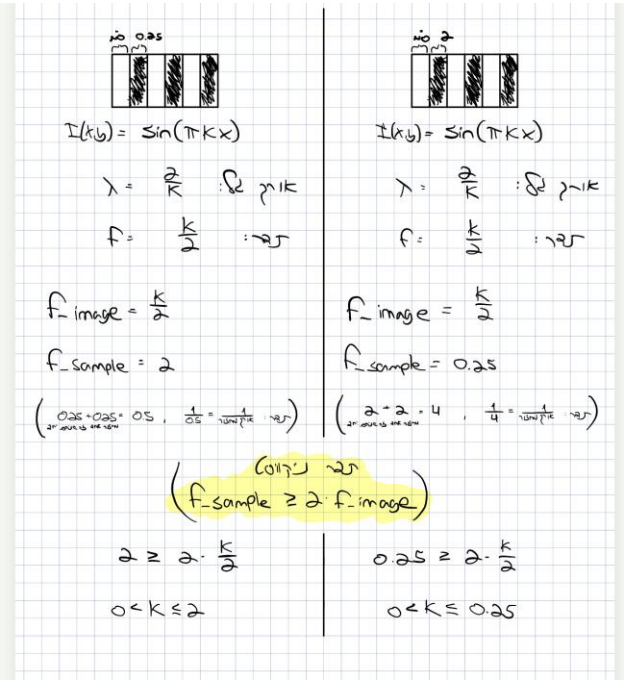
חומרה: שיקול ראשון הוא הגודל של המצלמה עצמה שמשפיע על כמות הפיקסלים שאפשר להכניס, בנוסף (גם עבור המחשב) עבור תמונות ברזולוציה גבוהה יותר (עם יותר פיקסלים) נצטרך מעבד חזק יותר,מקום אחסון גדול יותר (יותר דטה), מסך עם רזולוציה תואמת, דברים שהם יקרים יותר אז הפן הכלכלי הוא גם שיקול.

תוכנה: כדי לעבד ו/או לדחוס תמונות ברזולוציה גבוהה יותר נצטרך אלגוריתם יעיל יותר מאשר תמונה ברזולוציה נמוכה.

ב. שיקולים להחליט כמה חזק יהיה החישוב של התמונה? חישוב חזק יותר דורש מעבד חזק יותר ומקום אחסון גדול יותר, אך תורם בדחיסת התמונה באופן יעיל יותר, ובהצגת התמונה באופן ריאלי יותר.

בעבר מחשבים היו גדולים יותר, כי היה דרוש הרבה מקום אחסון, בעיקר כי החישוב היה חלש ולא הצליח לדחוס תמונות באופן יעיל כמו היום.

בנוסף מסכים של פעם לא תמכו ברזולוציות כמו היום לכן לא היה טעם לחישובים חזקים מדי כשהצג מוגבל לצבעים בודדים (תמונות שחור לבן.)




Problem2:

א. עבור 2 הערכים של A אורך הגל זהה:.$\frac{2}{K}$

ב. לפי משפט נייקוויסט-

עבור A=0.25 נקבל.k<=2

עבור A=2 נקבל.k<=0.25

)כמובן בשני המקרים גם k גדול מ0 .)

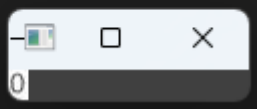
Problem3:

א. הוספת ה pathלתיקייה data לקוד.

ב. קריאת המספרים מהתיקיה:

```
cv2.imshow('names[0]', images[0])
cv2.waitKey(0)
cv2.destroyAllWindows()

# read digits
cv2.imshow('_[0]', numbers[0])
cv2.waitKey(0)
cv2.destroyAllWindows()
exit()
```




ג. יש לנו תמונה גדולה, וטרגט בחלון קטן יותר. נשתמש בחלון זז בגודל של הטרגט שיזוז על התמונה הגדולה באזור המתאים (המתרגל הציע אזור בפורום), נחשב היסטוגרמת הצטברות של כל חלון בעזרת הפונקציות המוצעות ונחשב .EMD אם עבור חלון כלשהו מצאנו התאמה נחזיר כן, אחרת נחזיר לא.

)צריך לסכום את כל ההפרשים המוחלטים אחרת נקבל שהתנאי שלנו יהיה if emd<260 return true כאשר emd הוא בעצם מערך של הפרשים, מה שכמובן לא יעבוד טוב(

נבדוק שהפונקציה עובדת באופן ידני על כל אחת מהתמונות. להלן כמה צילומי מסך:

```
Checking image a for all numbers from 0 to 9 (should be true for 6):
False
False
False
False
False
False
True
False
False
False
```

```
Checking image e (should be 5):
0 :  False
1 :  False
2 :  False
3 :  False
4 :  False
5 :  True
6 :  False
7 :  False
8 :  False
9 :  False
```

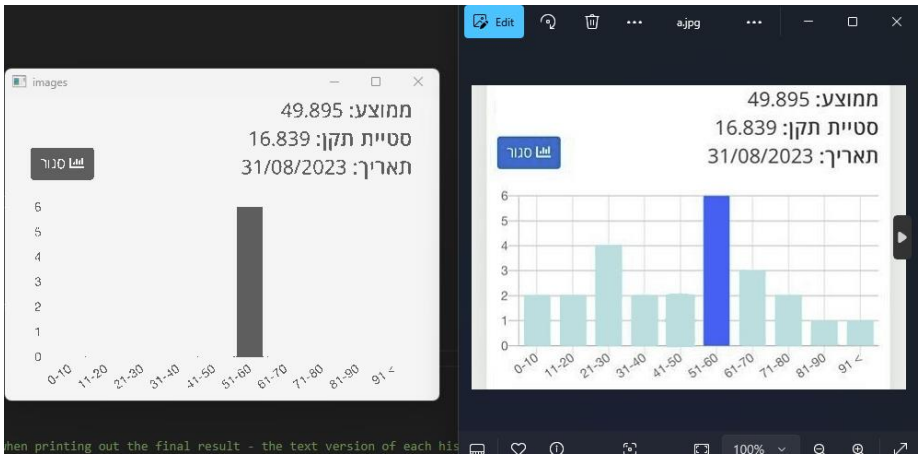
ככה בדקנו את כל התמונות וראינו שהפונקציה עובדת כמו שצריך.

```
for j in range (0,7):
    for i in range (9, -1 , -1):
        if compare_hist(images[j], numbers[i]):
            print("iamge", j, "gets result",i)
            break  # Exit the loop if the histograms match
```
ד.

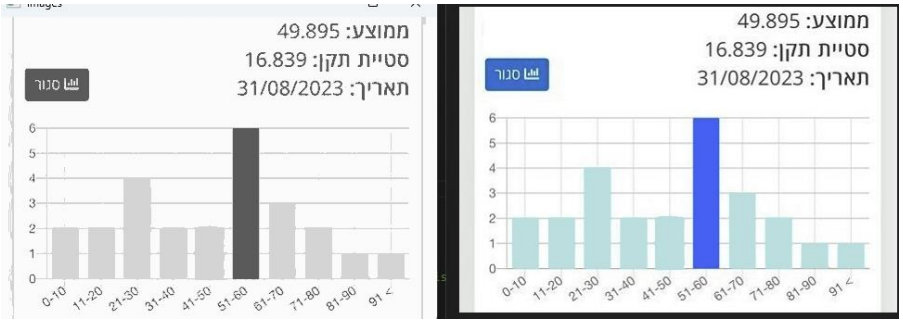
```
iamge 0 gets result 6
iamge 1 gets result 6
iamge 2 gets result 1
iamge 3 gets result 6
iamge 4 gets result 5
iamge 5 gets result 4
iamge 6 gets result 9
```

ה. נרצה שיהיו לנו רק 2 3 או 4 דרגות אפור וניעזר בפונקציה שהבאתם לנו, כדי שלא נצטרך לבדוק יותר מדי צבעים בתמונה.

עבור 2 צבעים בלבד אפשר לראות שאנחנו מאבדים נתונים עבור שאר הברים לכן זה לא מתאים:




ננסה 3 צבעים ונראה שזה עובד כמו שצריך, אפשר לזהות את הברים:



נהפוך לשחור או לבן אז נבדוק את הצבעים של כל התמונות כדי להבין מה ה thresholdהרצוי בעזרת

```
plt.imshow(quantized_images[0], cmap='gray')
plt.show()
```

מצאנו שהעמודות הבהירות הן בערכים 213,211,218,212,215

והעמודות בצבע כחול הן בערכים 90, 87, 91, 86, 92

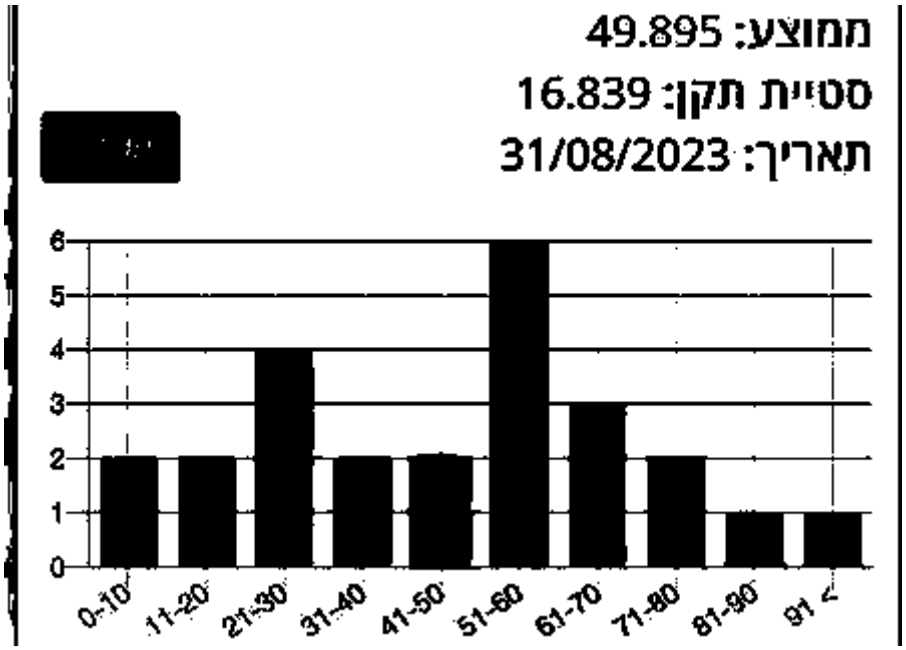
לכן ה thresholdשלנו יהיה 211-218 או 86-92 וכל הערכים שבטווח יהפכו ל0, והשאר יהיו 255

```
binary_img = np.full_like(quantized_images[0], 255,dtype='uint8')
binary_img[((quantized_images[0] >= 211) & (quantized_images[0]<=218))] = 0

plt.imshow(binary_img, cmap='gray')
plt.show()
```

)בסוף התרגיל ראינו שאם אנחנו מריצים כמה פעמים פעם אחר פעם אנחנו לפעמים מקבלים תוצאות שגויות, ושעדיף ללכת על טווח יותר רגיש לשינויים קטנים, לכן שינינו את ה threshhold למתחת 220 וכעת התוצאות עקביות ונכונות.(

```
for img in quantized_images:  # turn quantized_images to black and white
    binary_img = np.full_like(img, 255,dtype='uint8')  # start with an image all white
    binary_img[img<220] = 0  # for all pixels under 220 in image turn them black in binary image
    black_white_images.append(binary_img)  # add to black_white_images
```




ו. נחשב גובה של כל העמודות עבור כל תמונה ונשמור את המקסימום במערך max_bin_height

את המערך max student num יש לנו מ compare hist

Bin height פשוט נשתמש בפונקציה get height ובעזרת הנוסחה הנתונה בשאלה נקבל:

ז. כנדרש.

```
Histogram a.jpg gave [2, 2, 4, 2, 2, 6, 3, 2, 1, 1]
Histogram b.jpg gave [6, 2, 1, 1, 3, 3, 6, 2, 2, 3]
Histogram c.jpg gave [0, 0, 0, 0, 0, 0, 1, 1, 1, 1]
Histogram d.jpg gave [1, 0, 2, 3, 4, 3, 5, 5, 6, 2]
Histogram e.jpg gave [2, 1, 1, 3, 2, 5, 1, 1, 2, 3]
Histogram f.jpg gave [1, 0, 1, 1, 1, 4, 1, 1, 2, 1]
Histogram g.jpg gave [1, 1, 1, 3, 1, 2, 9, 3, 3, 0]
```