**SUMMARY REPORT**

**QUESTION 2 : Digital Image Creation (non Overlapping Rectangles) using**

**Python.**

Library Used : 🡪 Numpy

🡪CV2

Description of variables Used:

M x N -- Is the image size.

Border -- Black Border of thickness (Border).

n ---- Number of non-overlapping rectangles.

W1 to w2 -- width uniformly distributed in that range for rectangles.

Alpha -- rectangles height to width ratio.

Orientation -- landscape (1) or portrait (2).

Vf and vb -- foreground and background intensity (optional).

M\_total , N\_total – Final dimension after adding border.

M\_total = M + 2 \* (border)  
N\_total = N + 2 \* (border)

Corner\_i , corner\_j --- Corner pixel point of the rectangles to be drawn.

Discussion related To function Used in our code:

def valid\_rectangle(image\_bg, M\_total, N\_total, height, width, border, Vb, rect\_orientation, corner\_i, corner\_j):

🡪Above Function is Used in our code to check that the new rectangle drawn is not overlapping with other rectangles.

🡪Under this function code for both landscape and portrait orientation is written.

🡪This function is recursively called to find out non overlapping space for the new rectangle.

def create\_rectangles(M, N, border, n, w1, w2, alpha, orientation, Vf=[0], Vb=[255]):

🡪This function is used in our code to draw valid rectangles and create final image using cv2.imshow().

🡪vf and vb values are passed in function argument itself in case if Vf(black) and Vb(white) value are not passed by the user.

🡪In case of recursion error that is condition for non-overlapping rectangles is not satisfied we increase the size to (2\*M) x (2\*N).

def main():

🡪In main () function all the value of parameter is passed and under this function only create\_rectangles function is called.

**OUTPUT ANALYSIS**

CASE 1 : All the parameter is passed along with Vf [0,128] and vb [129,255]

**INPUT**

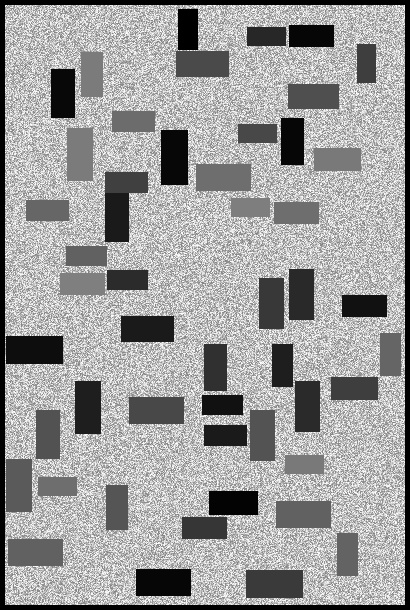
M = 300 # no of rows  
N = 200 # no of columns  
border = 5 # size of the border  
n = 50 # number of rectangles to fit  
w1 = 20 # lower bound for the width of rectangle  
w2 = 30 # upper bound for the width of rectangle  
alpha = 2 # fixed [height,width] ratio of rectangle  
orientation = [1, 2] # uniformly distributed orientation of rectangle [portrait, landscape]  
Vf = [0, 128] # foreground colours distributed uniformly, if not provided then default = 0  
Vb = [129, 255] # background colours distributed uniformly, if not provided then default = 255

**Function Call**

create\_rectangles(M, N, border, n, w1, w2, alpha, orientation, Vf, Vb)

**OUTPUT CASE 1:**

final shape of the image: (610, 410)



* From this output we can see all the rectangles are non-overlapping.
* Rectangles colour intensity is uniformly distributed as passed for Vf and also same case for background

CASE 2 : All the parameter is same as previous case but without Vf and Vb values being passed.

**INPUT**

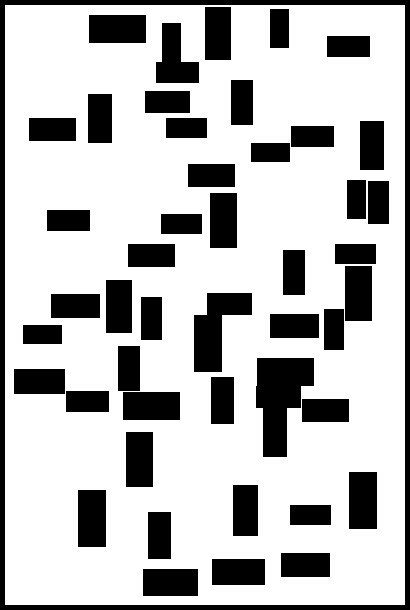
M = 300 # no of rows  
N = 200 # no of columns  
border = 5 # size of the border  
n = 50 # number of rectangles to fit  
w1 = 20 # lower bound for the width of rectangle  
w2 = 30 # upper bound for the width of rectangle  
alpha = 2 # fixed [height,width] ratio of rectangle  
orientation = [1, 2] # uniformly distributed orientation of rectangle [portrait, landscape]

**Function Call:**

create\_rectangles(M, N, border, n, w1, w2, alpha, orientation)

**OUTPUT Case 2:**

**Final shape of the image: (610, 410)**

****

* From this image it is clear when Vf and Vb values are not provided we get non -overlapping rectangles with vf=0 and vb=255 along with black border as coded.