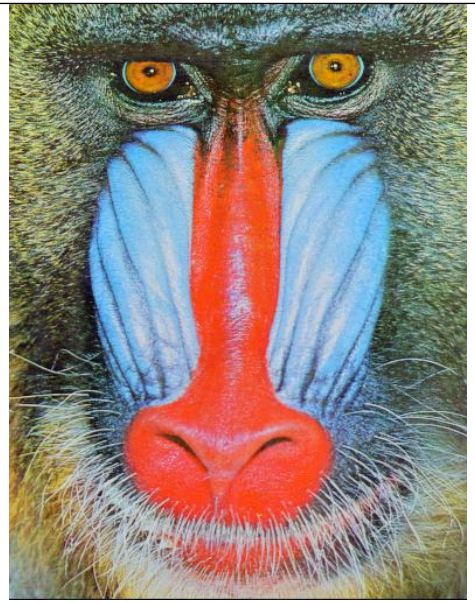
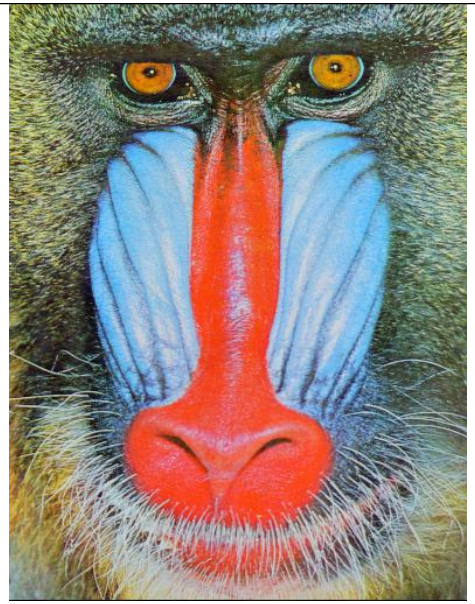
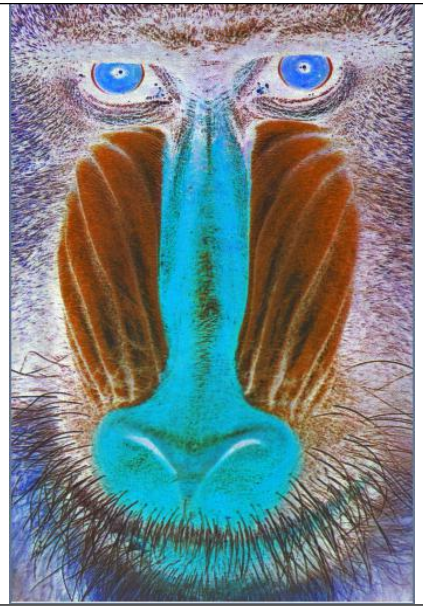
Form Evaluasi Project Tahap Akhir (UAS)

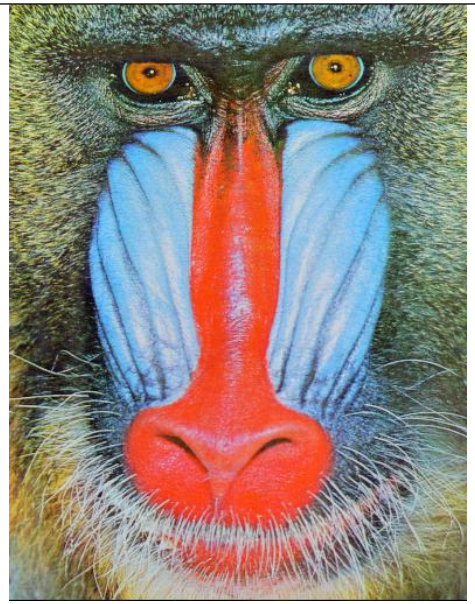
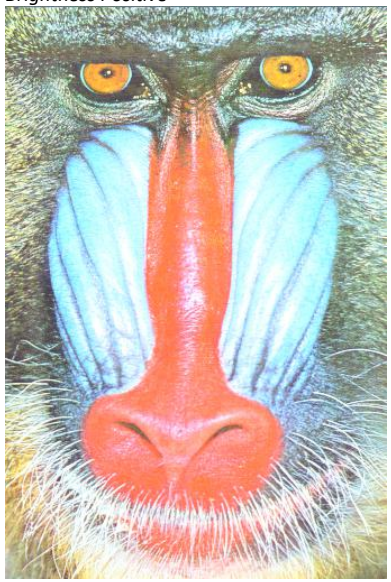
NAMA : IVANA PUSPITA SARI
NIM : 2110410278
KELAS : E

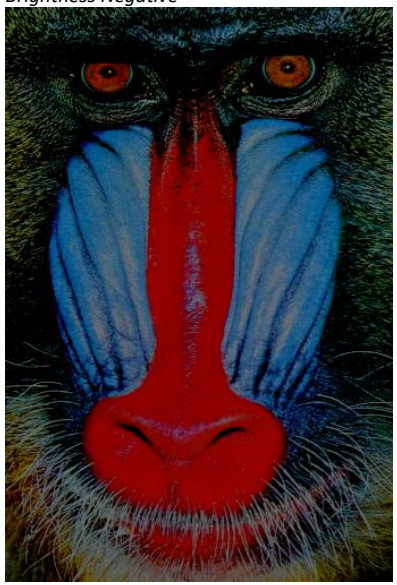
FITUR DASAR :

No	Fitur	Ada/ Tidak	Code Fungsi / Algoritma Utama	Image Input	Image Output

1	Image Thresholding	Ada	def ImgThresholding(img_input,coldepth):
			
			if coldepth!=24:
			img_input = img_input.convert('RGB')
			
			img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			pixels = img_output.load()
			thresholdpic = img_input.load()
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			if thresholdpic[i,j] < (127,127,127):
			pixels[i,j] = (0, 0, 0)
			elif thresholdpic[i,j] >= (127,127,127):
			pixels[i,j] = (255, 255, 255)
			
			if coldepth==1:
			img_output = img_output.convert("1")
			elif coldepth==8:
			img_output = img_output.convert("L")
			else:
			img_output = img_output.convert("RGB")
			
			return img_output
					

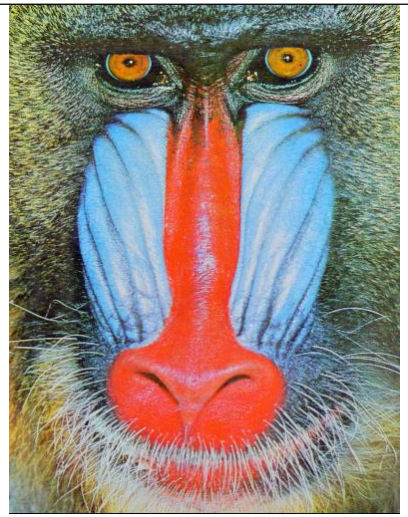
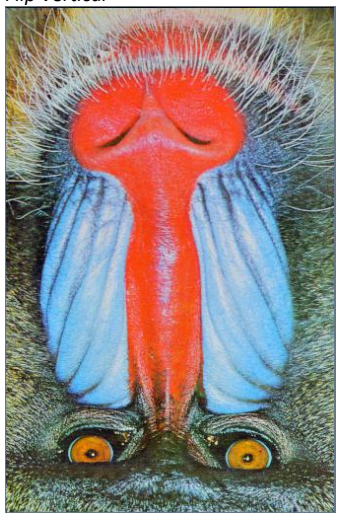
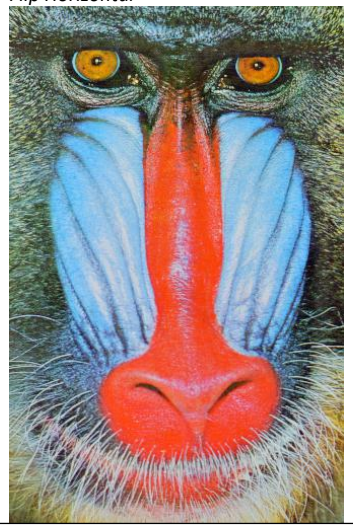
2	Image Negative	Ada	def ImgNegative(img_input,coldepth):
			
			  if coldepth!=24:
			  img_input = img_input.convert('RGB')
			
			  img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			  pixels = img_output.load()
			  for i in range(img_output.size[0]):
			  for j in range(img_output.size[1]):
			  r, g, b = img_input.getpixel((i, j))
			  pixels[i,j] = (255-r, 255-g, 255-b)
			
			  if coldepth==1:
			  img_output = img_output.convert("1")
			  elif coldepth==8:
			  img_output = img_output.convert("L")
			  else:
			  img_output = img_output.convert("RGB")
			
			  return img_output
					

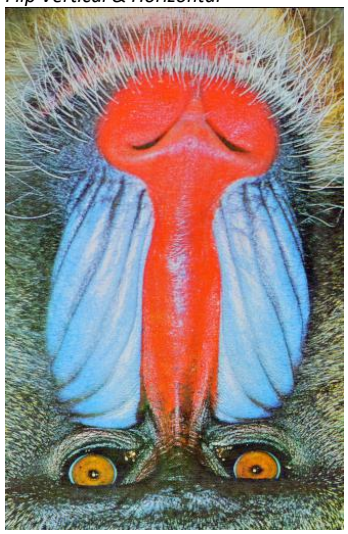
3	Image Brightness	Ada	#Brightness Posotive
			def ImgBrightnessPos(img_input,coldepth):
			  if coldepth!=24:
			  img_input = img_input.convert('RGB')
			
			  img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			  pixels = img_output.load()
			  for i in range(img_output.size[0]):
			  for j in range(img_output.size[1]):
			  r, g, b = img_input.getpixel((i, j))
			  pixels[i,j] = (r+60, g+60, b+60)
			  #cliping
			  if(r<0 and g<0 and b<0):
			  r=0
			  g=0
			  b=0
			
			  if coldepth==1:
			  img_output = img_output.convert("1")
			  elif coldepth==8:
			  img_output = img_output.convert("L")
			  else:
			  img_output = img_output.convert("RGB")
					*Brightness Positive*
					 

			return img_output
			
			#Brightness Negative
			def ImgBrightnessNeg(img_input,coldepth):
			if coldepth!=24:
			img_input = img_input.convert('RGB')
			
			img_output =
			Image.new('RGB',(img_input.size[0],img_input.size[1]))
			pixels = img_output.load()
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			r, g, b = img_input.getpixel((i, j))
			pixels[i,j] = (r-100, g-100, b-100)
			#cliping
			if(r<0 and g<0 and b<0):
			r=0
			g=0
			b=0
			
			if coldepth==1:
			img_output = img_output.convert("1")
			elif coldepth==8:
			img_output = img_output.convert("L")
			else:
			img_output = img_output.convert("RGB")
			
			return img_output
					Brightness Negative
					 

4	Image Rotation	Ada	def ImgRotate(img_input, coldepth, degree):
			if coldepth != 24:
			img_input = img_input.convert('RGB')
			
			pixels = img_input.load()
			
			img_output = Image.new('RGB', img_input.size)
			Draw = ImageDraw.Draw(img_output)
			
			sudut = radians(degree)
			Xcenter = img_input.width/2
			Ycenter = img_input.height/2
			
			for x in range(img_input.width):
			for y in range(img_input.height):
			XP = int((x - Xcenter) * cos(sudut) - (y - Ycenter) *
			sin(sudut) + Xcenter)
			YP = int((x - Xcenter) * sin(sudut) + (y - Ycenter) *
					

			cos(sudut) + Ycenter) if 0 <= XP < img_input.width and 0 <= YP < img_input.height: Draw.point((x, y), pixels[XP, YP]) return img_output		

5	Image Flipping	Ada	def ImgFlipVertical(img_input, coldepth):
			
			if coldepth != 25:
			img_input = img_input.convert('RGB')
			
			img_output = Image.new('RGB', (img_input.size[1], img_input.size[0]))
			pixels = img_output.load()
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			r, g, b = img_input.getpixel((i, img_input.size[1] - 1 - j))
			pixels[i, j] = (r, g, b)
			
			if coldepth == 1:
			img_output = img_output.convert("1")
			elif coldepth == 8:
			img_output = img_output.convert("L")
			else:
			img_output = img_output.convert("RGB")
			
			return img_output
			
			def ImgFlipHorizontal(img_input, coldepth):
			
			if coldepth != 25:
			img_input = img_input.convert('RGB')
			
			img_output = Image.new('RGB', (img_input.size[1], img_input.size[0]))
			pixels = img_output.load()
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			r, g, b = img_input.getpixel((img_input.size[0] - 1 - i, j))
			pixels[i, j] = (r, g, b)
			
			if coldepth == 1:
			img_output = img_output.convert("1")
			elif coldepth == 8:
			img_output = img_output.convert("L")
			else:
			img_output = img_output.convert("RGB")
					*Flip Vertical*
					 
					Flip Horizontal
					 

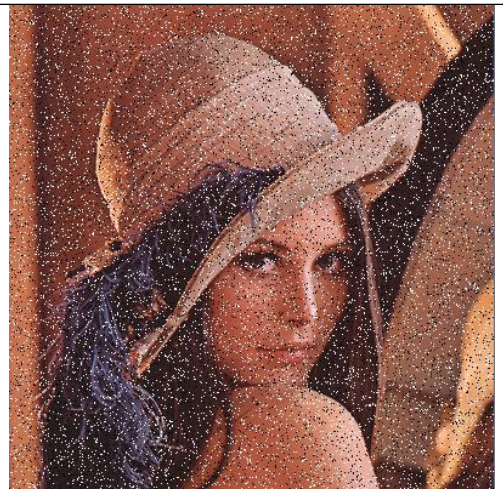
			return img_output
			
			def ImgFlipVerti_Hori(img_input, coldepth):
			
			if coldepth != 25:
			img_input = img_input.convert('RGB')
			
			img_output = Image.new('RGB', (img_input.size[1], img_input.size[0]))
			pixels = img_output.load()
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			r, g, b = img_input.getpixel((img_input.size[0] - 1 - i, img_input.size[0] - 1 - j))
			pixels[i, j] = (r, g, b)
			
			if coldepth == 1:
			img_output = img_output.convert("1")
			elif coldepth == 8:
			img_output = img_output.convert("L")
			else:
			img_output = img_output.convert("RGB")
			
			return img_output
					Flip Vertical & Horizontal
					 

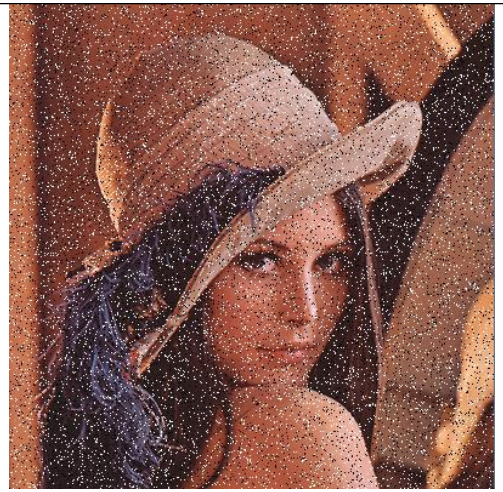
6	Image Zooming	Ada	def ImgZooming(img_input, coldepth, skala):
			if coldepth != 24:
			img_input = img_input.convert('RGB')
			if skala == 2:
			N = 2
			img_output = Image.new('RGB', (img_input.size[0]*N, img_input.size[1]*N))
			pixels = img_output.load()
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			r, g, b = img_input.getpixel((int(i/N), int(j/N)))
			pixels[i, j] = (r, g, b)
			elif skala == 3:
			N = 3
			img_output = Image.new('RGB', (img_input.size[0]*N, img_input.size[1]*N))
			pixels = img_output.load()
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			r, g, b = img_input.getpixel((int(i/N), int(j/N)))
			pixels[i, j] = (r, g, b)
			elif skala == 4:
					

			N = 4
			img_output = Image.new('RGB', (img_input.size[0]*N, img_input.size[1]*N))
			pixels = img_output.load()
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			r, g, b = img_input.getpixel((int(i/N), int(j/N)))
			pixels[i, j] = (r, g, b)
			
			if coldepth == 1:
			img_output = img_output.convert("1")
			elif coldepth == 8:
			img_output = img_output.convert("L")
			else:
			img_output = img_output.convert("RGB")
			
			return img_output		

7	Image Shrinking	Ada	def ImgShrinking(img_input, coldepth, skala):
			if coldepth != 24:
			img_input = img_input.convert('RGB')
			if skala == 2:
			N = 2
			img_output = Image.new('RGB', (int(img_input.size[0]/N), int(img_input.size[1]/N)))
			pixels = img_output.load()
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			r, g, b = img_input.getpixel((i*N, j*N))
			pixels[i, j] = (r, g, b)
			elif skala == 3:
			N = 3
			img_output = Image.new('RGB', (int(img_input.size[0]/N), int(img_input.size[1]/N)))
			pixels = img_output.load()
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			r, g, b = img_input.getpixel((i*N, j*N))
			pixels[i, j] = (r, g, b)
			elif skala == 4:
			N = 4
			img_output = Image.new('RGB', (int(img_input.size[0]/N), int(img_input.size[1]/N)))
			pixels = img_output.load()
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			r, g, b = img_input.getpixel((i*N, j*N))
			pixels[i, j] = (r, g, b)
					

			if coldepth == 1:
			img_output = img_output.convert("1")
			elif coldepth == 8:
			img_output = img_output.convert("L")
			else:
			img_output = img_output.convert("RGB")
			
			return img_output		

8	Mean Filtering	Ada	def ImgMeanFilter(img_input, coldepth):
			if coldepth != 25:
			img_input = img_input.convert("RGB")
			pixels = img_input.load()
			img_output = Image.new(
			'RGB', (img_input.size[1], img_input.size[0]))
			output_pixels = img_output.load()
			
			box_kernel = [
			[1 / 9, 1 / 9, 1 / 9],
			[1 / 9, 1 / 9, 1 / 9],
			[1 / 9, 1 / 9, 1 / 9]]
			
			kernel = box_kernel
			offset = len(kernel)//2
			
			for x in range(offset, img_input.width - offset):
			for y in range(offset, img_input.height - offset):
			acc = [0,0,0]
			for a in range(len(kernel)):
			for b in range(len(kernel)):
			xn = x + a - offset
			yn = y + b - offset
			pixel = pixels[xn, yn]
			acc[0] += pixel[0] * kernel[a][b]
			acc[1] += pixel[1] * kernel[a][b]
			acc[2] += pixel[2] * kernel[a][b]
			output_pixels[x,y] = (int(acc[0]), int(acc[1]), int(acc[2]))
			if coldepth == 1:
			img_output = img_output.convert("1")
			elif coldepth == 8:
			img_output = img_output.convert("L")
			else:
			img_output = img_output.convert("RGB")
			
			return img_output
					

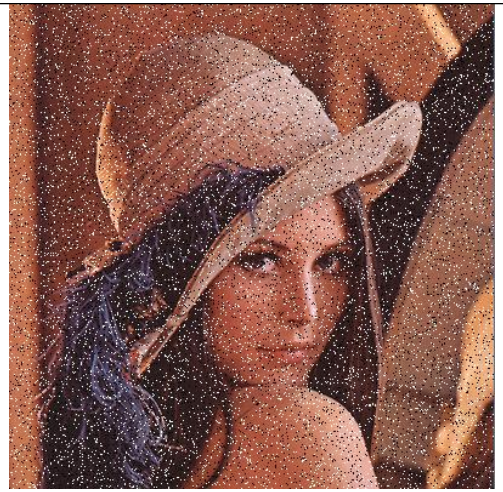
9	Median Filtering	Ada	def ImgMedianFilter(img_input, color_depth):
			
			if color_depth != 24:
			img_input = img_input.convert("RGB")
			output_image = Image.new("RGB", (img_input.size[0], img_input.size[1]), "white")
			# output_pixels = output_image.load()
			mask = [(0, 0)] * 9
			for i in range(img_input.size[0]-1):
			for j in range(img_input.size[1]-1):
			mask[0] = img_input.getpixel((i-1, j-1))
			mask[1] = img_input.getpixel((i-1, j))
			mask[2] = img_input.getpixel((i-1, j+1))
			
			mask[3] = img_input.getpixel((i, j-1))
			mask[4] = img_input.getpixel((i, j))
			mask[5] = img_input.getpixel((i, j+1))
			
			mask[6] = img_input.getpixel((i+1, j-1))
			mask[7] = img_input.getpixel((i+1, j))
			mask[8] = img_input.getpixel((i+1, j+1))
			mask.sort()
			output_image.putpixel((i, j), (mask[4]))
			
			if color_depth == 1:
			output_image = output_image.convert("1")
			elif color_depth == 8:
			output_image = output_image.convert("L")
			else:
			output_image = output_image.convert("RGB")
			
			return output_image
					

10	Edge Detection Pilihan 1 (Laplacian)	Ada			

```
def ImgEdgeDetection(img_input, coldepth, val, menu):

    img_input = img_input.convert('L')

    img_output =
Image.new('L',(img_input.size[0],img_input.size[1]))
    pixels = img_output.load()

    for i in range(img_output.size[0]):
        for j in range(img_output.size[1]):
            if i > img_output.size[0]-3 or j > img_output.size[1]-3 :
                pixels[i,j] = img_input.getpixel((i, j))
            else:
                p1 = img_input.getpixel((i, j))
                p2 = img_input.getpixel((i, j+1))
                p3 = img_input.getpixel((i, j+2))
                p4 = img_input.getpixel((i+1, j))
                p5 = img_input.getpixel((i+1, j+1))
                p6 = img_input.getpixel((i+1, j+2))
                p7 = img_input.getpixel((i+2, j))
                p8 = img_input.getpixel((i+2, j+1))
                p9 = img_input.getpixel((i+2, j+2))

                if menu==1: #Laplacian
                    x = abs(p2+p4+p5*(-4)+p6+p8)
                    y = x
                elif menu==2: #Prewitt
                    x = abs(p1*(-1)+p3+p4*(-1)+p6+p7*(-1)+p9)
                    y = abs(p1+p2+p3+p7*(-1)+p8*(-1)+p9*(-1))
                elif menu==3: #Robert
                    x = abs(p1+p5*(-1))
                    y = abs(p2+p4*(-1))
                elif menu==4: #Sobel
                    x = abs(p1+p3*(-1)+p4*2+p6*(-2)+p7+p9*(-1))
                    y = abs(p1+p2*2+p3+p7*(-1)+p8*(-2)+p9*(-1))

                else:
                    x=p1
                    y=p1
                xy = x+y

                if xy<val :
                    pixels[i,j] = (0)
                else:
                    pixels[i,j] = (255)
    img_output = img_output.convert("L")
    return img_output
```

11	Edge Detection Pilihan 2 (Prewitt)	Ada			

```
def ImgEdgeDetection(img_input, coldepth, val, menu):

    img_input = img_input.convert('L')

    img_output =
Image.new('L',(img_input.size[0],img_input.size[1]))
    pixels = img_output.load()

    for i in range(img_output.size[0]):
        for j in range(img_output.size[1]):
            if i > img_output.size[0]-3 or j > img_output.size[1]-3 :
                pixels[i,j] = img_input.getpixel((i, j))
            else:
                p1 = img_input.getpixel((i, j))
                p2 = img_input.getpixel((i, j+1))
                p3 = img_input.getpixel((i, j+2))
                p4 = img_input.getpixel((i+1, j))
                p5 = img_input.getpixel((i+1, j+1))
                p6 = img_input.getpixel((i+1, j+2))
                p7 = img_input.getpixel((i+2, j))
                p8 = img_input.getpixel((i+2, j+1))
                p9 = img_input.getpixel((i+2, j+2))

                if menu==1: #Laplacian
                    x = abs(p2+p4+p5*(-4)+p6+p8)
                    y = x
                elif menu==2: #Prewitt
                    x = abs(p1*(-1)+p3+p4*(-1)+p6+p7*(-1)+p9)
                    y = abs(p1+p2+p3+p7*(-1)+p8*(-1)+p9*(-1))
                elif menu==3: #Robert
                    x = abs(p1+p5*(-1))
                    y = abs(p2+p4*(-1))
                elif menu==4: #Sobel
                    x = abs(p1+p3*(-1)+p4*2+p6*(-2)+p7+p9*(-1))
                    y = abs(p1+p2*2+p3+p7*(-1)+p8*(-2)+p9*(-1))

                else:
                    x=p1
                    y=p1
                xy = x+y

                if xy<val :
                    pixels[i,j] = (0)
                else:
                    pixels[i,j] = (255)
    img_output = img_output.convert("L")
    return img_output
```

12	Gaussian Filtering	Ada	def ImgGaussian(img_input,coldepth):
			
			  if coldepth!=24:
			  img_input = img_input.convert('RGB')
			  image_input = Image.new('RGB',(img_input.size[0]+4,img_input.size[1]+4))
			  pix = image_input.load()
			  img_output = Image.new('RGB',(img_input.size[0],img_input.size[1]))
			  pixels = img_output.load()
			
			  for i in range(image_input.size[0]):
			  for j in range(image_input.size[1]):
			
			  if i<2 or j<2 or i>= image_input.size[0]-2 or j>= image_input.size[1]-2:
			  pix[i,j] = (0, 0, 0)
			  else:
			  r, g, b = img_input.getpixel((i-2,j-2))
			  pix[i,j] = (r, g, b)
			
			  for i in range(img_input.size[0]):
			  for j in range(img_input.size[1]):
			  r0,g0,b0=(image_input.getpixel((i+1,j+1)))
			  r1,g1,b1=(image_input.getpixel((i+1,j+2)))
			  r2,g2,b2=(image_input.getpixel((i+1,j+3)))
			
			  r3,g3,b3=(image_input.getpixel((i+2,j+1)))
			  r4,g4,b4=(image_input.getpixel((i+2,j+2)))
			  r5,g5,b5=(image_input.getpixel((i+2,j+3)))
			
			  r6,g6,b6=(image_input.getpixel((i+3,j+1)))
			  r7,g7,b7=(image_input.getpixel((i+3,j+2)))
			  r8,g8,b8=(image_input.getpixel((i+3,j+3)))
			
			  r=int((r0+r1*2+r2+r3*2+r4*4+r5*2+r6+r7*2+r8)/16)
			  g=int((g0+g1*2+g2+g3*2+g4*4+g5*2+g6+g7*2+g8)/16)
			
			b=int((b0+b1*2+b2+b3*2+b4*4+b5*2+b6+b7*2+b8)/16)
			
			  pixels[i,j] = (r, g, b)
			
			  img_output = img_output.convert("RGB")
			
			  return img_output
					

13	Erosi	Ada	def ImgMorfologi(img_input, coldepth, menu):
			
			img_input = img_input.convert('L')
			
			image_input = Image.new('L',(img_input.size[0]+4,img_input.size[1]+4))
			pix = image_input.load()
			
			img_output = Image.new('L',(img_input.size[0],img_input.size[1]))
			pixels = img_output.load()
			
			for i in range(image_input.size[0]):
			for j in range(image_input.size[1]):
			
			if i<2 or j<2 or i>=image_input.size[0]-2 or j>=image_input.size[1]-2:
			pix[i,j] = (0)
			else:
			p= img_input.getpixel((i-2,j-2))
			pix[i,j] = (p)
			
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			
			p0=(image_input.getpixel((i+1,j+1)))
			p1=(image_input.getpixel((i+1,j+2)))
			p2=(image_input.getpixel((i+1,j+3)))
			
			p3=(image_input.getpixel((i+2,j+1)))
			p4=(image_input.getpixel((i+2,j+2)))
			p5=(image_input.getpixel((i+2,j+3)))
			
			p6=(image_input.getpixel((i+3,j+1)))
			p7=(image_input.getpixel((i+3,j+2)))
			p8=(image_input.getpixel((i+3,j+3)))
			
			p = [p0,p1,p2,p3,p4,p5,p6,p7,p8]
			
			for iterasi in range(len(p)-1,0,-1):
			for ind in range(iterasi):
			if p[ind] > p[ind+1]:
			temp = p[ind]
			p[ind] = p[ind+1]
			p[ind+1] = temp
			
			if menu==2: #2:dilasi
					

			q=p[0] else: q=p[len(p)-1] #1:erosi pixels[i,j] = (q) img_output = img_output.convert("L") return img_output		

14	Dilasi	Ada	def ImgMorfologi(img_input, coldepth, menu):
			
			img_input = img_input.convert('L')
			
			image_input = Image.new('L',(img_input.size[0]+4,img_input.size[1]+4))
			pix = image_input.load()
			
			img_output = Image.new('L',(img_input.size[0],img_input.size[1]))
			pixels = img_output.load()
			
			for i in range(image_input.size[0]):
			for j in range(image_input.size[1]):
			
			if i<2 or j<2 or i>=image_input.size[0]-2 or j>=image_input.size[1]-2:
			pix[i,j] = (0)
			else:
			p= img_input.getpixel((i-2,j-2))
			pix[i,j] = (p)
			
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			
			p0=(image_input.getpixel((i+1,j+1)))
			p1=(image_input.getpixel((i+1,j+2)))
			p2=(image_input.getpixel((i+1,j+3)))
			
			p3=(image_input.getpixel((i+2,j+1)))
			p4=(image_input.getpixel((i+2,j+2)))
			p5=(image_input.getpixel((i+2,j+3)))
			
			p6=(image_input.getpixel((i+3,j+1)))
			p7=(image_input.getpixel((i+3,j+2)))
			p8=(image_input.getpixel((i+3,j+3)))
			
			p = [p0,p1,p2,p3,p4,p5,p6,p7,p8]
					

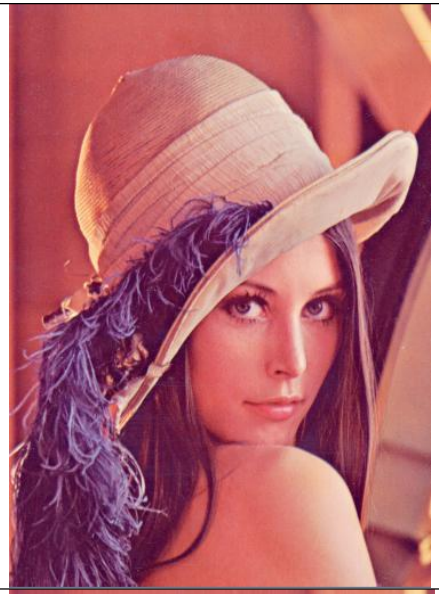
			for iterasi in range(len(p)-1,0,-1):
			for ind in range(iterasi):
			if p[ind] > p[ind+1]:
			temp = p[ind]
			p[ind] = p[ind+1]
			p[ind+1] = temp
			
			if menu==2: #2:dilasi
			q=p[0]
			else:
			q=p[len(p)-1] #1:erosi
			
			pixels[i,j] = (q)
			
			img_output = img_output.convert("L")
			
			return img_output		

15	Opening	Ada	def ImgMorfologi(img_input, coldepth, menu):
			
			img_input = img_input.convert('L')
			
			image_input = Image.new('L',(img_input.size[0]+4,img_input.size[1]+4))
			pix = image_input.load()
			
			img_output = Image.new('L',(img_input.size[0],img_input.size[1]))
			pixels = img_output.load()
			
			for i in range(image_input.size[0]):
			for j in range(image_input.size[1]):
			
			if i<2 or j<2 or i>=image_input.size[0]-2 or j>=image_input.size[1]-2:
			pix[i,j] = (0)
			else:
			p= img_input.getpixel((i-2,j-2))
			pix[i,j] = (p)
			
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			
			p0=(image_input.getpixel((i+1,j+1)))
			p1=(image_input.getpixel((i+1,j+2)))
			p2=(image_input.getpixel((i+1,j+3)))
					

			p3=(image_input.getpixel((i+2,j+1)))
			p4=(image_input.getpixel((i+2,j+2)))
			p5=(image_input.getpixel((i+2,j+3)))
			
			p6=(image_input.getpixel((i+3,j+1)))
			p7=(image_input.getpixel((i+3,j+2)))
			p8=(image_input.getpixel((i+3,j+3)))
			
			p = [p0,p1,p2,p3,p4,p5,p6,p7,p8]
			
			for iterasi in range(len(p)-1,0,-1):
			for ind in range(iterasi):
			if p[ind] > p[ind+1]:
			temp = p[ind]
			p[ind] = p[ind+1]
			p[ind+1] = temp
			
			if menu==2: #2:dilasi
			q=p[0]
			else:
			q=p[len(p)-1] #1:erosi
			
			pixels[i,j] = (q)
			
			img_output = img_output.convert("L")
			
			return img_output		

16	Closing	Ada	def ImgMorfologi(img_input, coldepth, menu):
			
			img_input = img_input.convert('L')
			
			image_input = Image.new('L',(img_input.size[0]+4,img_input.size[1]+4))
			pix = image_input.load()
			
			img_output = Image.new('L',(img_input.size[0],img_input.size[1]))
			pixels = img_output.load()
			
			for i in range(image_input.size[0]):
			for j in range(image_input.size[1]):
			
			if i<2 or j<2 or i>=image_input.size[0]-2 or j>=image_input.size[1]-2:
			pix[i,j] = (0)
			else:
			p= img_input.getpixel((i-2,j-2))
					

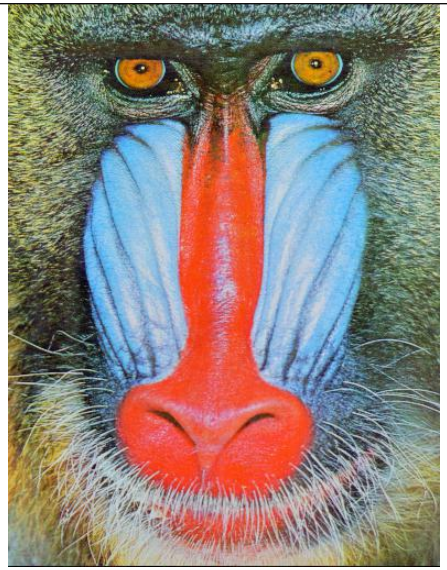
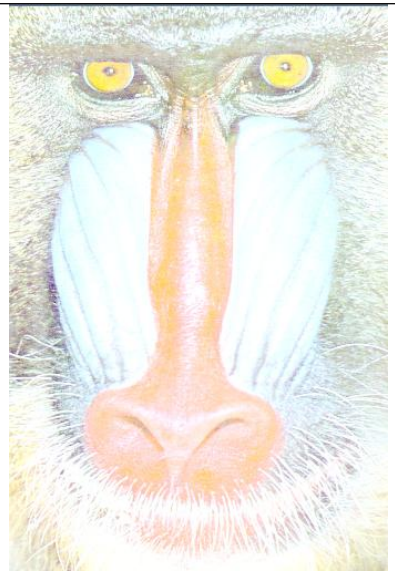
```
            pix[i,j] = (p)

    for i in range(img_output.size[0]):
        for j in range(img_output.size[1]):

            p0=(image_input.getpixel((i+1,j+1)))
            p1=(image_input.getpixel((i+1,j+2)))
            p2=(image_input.getpixel((i+1,j+3)))

            p3=(image_input.getpixel((i+2,j+1)))
            p4=(image_input.getpixel((i+2,j+2)))
            p5=(image_input.getpixel((i+2,j+3)))

            p6=(image_input.getpixel((i+3,j+1)))
            p7=(image_input.getpixel((i+3,j+2)))
            p8=(image_input.getpixel((i+3,j+3)))

            p = [p0,p1,p2,p3,p4,p5,p6,p7,p8]

            for iterasi in range(len(p)-1,0,-1):
                for ind in range(iterasi):
                    if p[ind] > p[ind+1]:
                        temp = p[ind]
                        p[ind] = p[ind+1]
                        p[ind+1] = temp

            if menu==2: #2:dilasi
                q=p[0]
            else:
                q=p[len(p)-1] #1:erosi

            pixels[i,j] = (q)

    img_output = img_output.convert("L")

    return img_output
```

17	RGB to Grayscale	Ada	def ImgRgb2Grayscale(input_image, coldepth):
			
			if coldepth!=24:
			input_image = input_image.convert('RGB')
			
			img_output = Image.new('RGB',(input_image.size[0],input_image.size[1]))
			pixels = img_output.load()
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			r, g, b = input_image.getpixel((i, j))
			pixels[i,j] = (r, r, r)
			
			if coldepth==1:
			img_output = img_output.convert("1")
			elif coldepth==8:
			img_output = img_output.convert("L")
			else:
			img_output = img_output.convert("RGB")
			
			return img_output
					

18	RGB to HSV/HLS Conversion	Ada	def ImgRgb2Hsv(img_input, coldepth):
			
			if coldepth!=24:
			img_input = img_input.convert('RGB')
			
			img_output = Image.new('L',(img_input.size[0],img_input.size[1]))
			pixels = img_output.load()
			
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			r, g, b= img_input.getpixel((i,j))
			p = max(r,g,b)
			pixels[i,j] = (p)
			
			img_output = img_output.convert("L")
			
			return img_output
					

FITUR TAMBAHAN :

No	Fitur	Ada/Tidak	Code Algoritma	Image Input	Image Output
1	Image Blending	Tidak			

2	Image Logarithmic	Ada	def ImgLogarithmic(img_input, coldepth):
			
			if coldepth != 25:
			img_input = img_input.convert('RGB')
			c = 50
			img_output = Image.new('RGB', (img_input.size[0], img_input.size[1]))
			pixels = img_output.load()
			
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			r, g, b = img_input.getpixel((i, j))
			pixels[i, j] = (int(c*math.log(1+r)), int(c*math.log(1+g)), int(c*math.log(1+b)))
			
			if coldepth == 1:
			img_output = img_output.convert("1")
			elif coldepth == 8:
			img_output = img_output.convert("L")
			else:
			img_output = img_output.convert("RGB")
			return img_output
					

3	Image Translation	Ada	def ImgTranslation(img_input, coldepth, shift):
			if coldepth != 25:
			img_input = img_input.convert("RGB")
			input_pixels = img_input.load()
			
			img_output = Image.new(
			'RGB', (img_input.size[1], img_input.size[0]))
			output_pixels = img_output.load()
			
			nilaii = shift[0]
			nilaij = shift[1]
			
			if shift[0] < 0:
			nilaii = 0
			if shift[1] < 0:
			nilaij = 0
			
			for x in range(nilaii, img_input.size[0]):
			for y in range(nilaij, img_input.size[1]):
			xbaru = x - shift[0]
					

```
            ybaru = y - shift[1]

            if(xbaru >= img_input.size[0] or ybaru >= img_input.size[1] or xbaru < 0 or ybaru < 0):
                output_pixels[x, y] = (0, 0, 0)
            else:
                output_pixels[x, y] = input_pixels[xbaru, ybaru]

    if coldepth == 1:
        img_output = img_output.convert("1")
    elif coldepth == 8:
        img_output = img_output.convert("L")
    else:
        img_output = img_output.convert("RGB")

    return img_output
```

4	Edge Detection Pilihan 3 (Robert)	Ada			

```
def ImgEdgeDetection(img_input, coldepth, val, menu):

    img_input = img_input.convert('L')

    img_output = Image.new('L',(img_input.size[0],img_input.size[1]))
    pixels = img_output.load()

    for i in range(img_output.size[0]):
        for j in range(img_output.size[1]):
            if i > img_output.size[0]-3 or j > img_output.size[1]-3 :
                pixels[i,j] = img_input.getpixel((i, j))
            else:
                p1 = img_input.getpixel((i, j))
                p2 = img_input.getpixel((i, j+1))
                p3 = img_input.getpixel((i, j+2))
                p4 = img_input.getpixel((i+1, j))
                p5 = img_input.getpixel((i+1, j+1))
                p6 = img_input.getpixel((i+1, j+2))
                p7 = img_input.getpixel((i+2, j))
                p8 = img_input.getpixel((i+2, j+1))
                p9 = img_input.getpixel((i+2, j+2))

                if menu==1: #Laplacian
                    x = abs(p2+p4+p5*(-4)+p6+p8)
                    y = x
                elif menu==2: #Prewitt
                    x = abs(p1*(-1)+p3+p4*(-1)+p6+p7*(-1)+p9)
                    y = abs(p1+p2+p3+p7*(-1)+p8*(-1)+p9*(-1))
                elif menu==3: #Robert
                    x = abs(p1+p5*(-1))
```

			y = abs(p2+p4*(-1))
			elif menu==4: #Sobel
			x = abs(p1+p3*(-1)+p4*2+p6*(-2)+p7+p9*(-1))
			y = abs(p1+p2*2+p3+p7*(-1)+p8*(-2)+p9*(-1))
			
			else:
			x=p1
			y=p1
			xy = x+y
			
			if xy<val :
			pixels[i,j] = (0)
			else:
			pixels[i,j] = (255)
			img_output = img_output.convert("L")
			return img_output		

5	Edge Detection Pilihan 4 (Sobel)	Ada	def ImgEdgeDetection(img_input, coldepth, val, menu):
			
			img_input = img_input.convert('L')
			
			img_output = Image.new('L',(img_input.size[0],img_input.size[1]))
			pixels = img_output.load()
			
			for i in range(img_output.size[0]):
			for j in range(img_output.size[1]):
			if i > img_output.size[0]-3 or j > img_output.size[1]-3 :
			pixels[i,j] = img_input.getpixel((i, j))
			else:
			p1 = img_input.getpixel((i, j))
			p2 = img_input.getpixel((i, j+1))
			p3 = img_input.getpixel((i, j+2))
			p4 = img_input.getpixel((i+1, j))
			p5 = img_input.getpixel((i+1, j+1))
			p6 = img_input.getpixel((i+1, j+2))
			p7 = img_input.getpixel((i+2, j))
			p8 = img_input.getpixel((i+2, j+1))
			p9 = img_input.getpixel((i+2, j+2))
			
			if menu==1: #Laplacian
			x = abs(p2+p4+p5*(-4)+p6+p8)
			y = x
			elif menu==2: #Prewitt
			x = abs(p1*(-1)+p3+p4*(-1)+p6+p7*(-1)+p9)
			y = abs(p1+p2+p3+p7*(-1)+p8*(-1)+p9*(-1))
			elif menu==3: #Robert
			x = abs(p1+p5*(-1))
					

			y = abs(p2+p4*(-1))
			elif menu==4: #Sobel
			x = abs(p1+p3*(-1)+p4*2+p6*(-2)+p7+p9*(-1))
			y = abs(p1+p2*2+p3+p7*(-1)+p8*(-2)+p9*(-1))
			
			else:
			x=p1
			y=p1
			xy = x+y
			
			if xy<val :
			pixels[i,j] = (0)
			else:
			pixels[i,j] = (255)
			img_output = img_output.convert("L")
			return img_output		

6	Edge Detection Pilihan 5	Tidak			
7	Top Hat	Tidak			

PRINT SCREEN ANTARMUKA UTAMA :