Raport 3 Jakub Jarmakowicz

1. Zadanie1

```
#zadnaie1
obraz=Image.open("im.png")
inicjaly=Image.open("inicjaly.bmp")
```

obraz:

Inicjaly:

2. Zadanie2
   a.

```
def wstaw_inicjaly(obraz, inicjaly, m, n, kolor):
    obraz=obraz.copy()

    h_b, w_b= obraz.size
    h_w, w_w = inicjaly.size

    n_p = max(0, n)
    m_p = max(0, m)
    n_k = min(h_b, n + h_w)
    m_k = min(w_b, m + w_w)

    for i in range(n_p, n_k):
        for j in range(m_p, m_k):
            if inicjaly.getpixel((i-n, j-m)) == 0:
                obraz.putpixel( (i , j) , kolor)
    return obraz
```

obraz1.png:

b.

```
def wstaw_inicjaly_maska(obraz, inicjaly, m, n):
    obraz = obraz.copy()
    h_b, w_b = obraz.size
    h_w, w_w = inicjaly.size

    n_p = max(0, n)
    m_p = max(0, m)
    n_k = min(h_b, n + h_w)
    m_k = min(w_b, m + w_w)

    for i in range(n_p, n_k):
        for j in range(m_p, m_k):
            if inicjaly.getpixel((i - n, j - m)) == 0:
                obraz.putpixel((i, j), (255-obraz.getpixel((i, j))[0],255-obraz.getpixel((i, j))[1],255-obraz.getpixel((i, j))[2]))
    return obraz
```

obraz2.png:

3. Zadanie3

```
def wstaw_inicjaly_load(obraz, inicjaly, m, n,kolor):
    obraz = obraz.copy()
    w_b, h_b= obraz.size
    w_w, h_w = inicjaly.size

    n_p = max(0, n)
    m_p = max(0, m)
    n_k = min(h_b, n + h_w)
    m_k = min(w_b, m + w_w)

    pix_obraz = obraz.load()
    pix_inicjaly = inicjaly.load()

    for i in range(n_p, n_k):
        for j in range(m_p, m_k):
            if pix_inicjaly[j-m,i-n] == 0:
                pix_obraz[j,i]=kolor
    return obraz
```

```
def wstaw_inicjaly_maska_load(obraz,inicjaly,m,n,x,y,z):
    obraz = obraz.copy()
    w_b, h_b = obraz.size
    w_w, h_w = inicjaly.size

    n_p = max(0, n)
    m_p = max(0, m)
    n_k = min(h_b, n + h_w)
    m_k = min(w_b, m + w_w)

    pix_obraz = obraz.load()
    pix_inicjaly = inicjaly.load()

    for i in range(n_p, n_k):
        for j in range(m_p, m_k):
            if pix_inicjaly[j-m,i-n] == 0:
                r,g,b = pix_obraz[j, i]
                pix_obraz[j, i]=(r-x, g-y, b-z)
    return obraz
```

fig1.png:

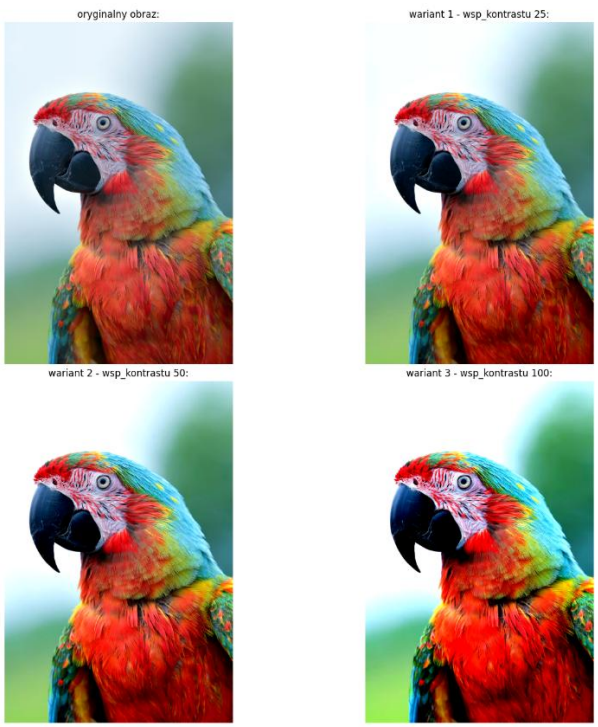
4. Zadanie4
   a.

```
def kontrast(obraz, wsp_kontrastu):  3 usages
    if wsp_kontrastu < 0 or wsp_kontrastu > 100:
        return None
    obraz = obraz.copy()
    mn = ((255+wsp_kontrastu)/255)**2
    return obraz.point(lambda i: 128+(i-128)*mn)

org=obraz
war1=kontrast(obraz, wsp_kontrastu: 25)
war2=kontrast(obraz, wsp_kontrastu: 50)
war3=kontrast(obraz, wsp_kontrastu: 100)
```

```
plt.figure(figsize=(16, 16))
plt.subplot( *args: 2,2,1) # ile obrazów w pionie, ile w poziomie, numer obrazu
plt.imshow(org)
plt.title("oryginalny obraz:")
plt.axis('off')
plt.subplot( *args: 2,2,2)
plt.title("wariant 1 - wsp_kontrastu 25:")
plt.imshow(war1)
plt.axis('off')
plt.subplot( *args: 2,2,3)
plt.title("wariant 2 - wsp_kontrastu 50:")
plt.imshow(war2)
plt.axis('off')
plt.subplot( *args: 2,2,4)
plt.title("wariant 3 - wsp_kontrastu 100:")
plt.imshow(war3)
plt.axis('off')
plt.subplots_adjust(wspace=0.05, hspace=0.05)
plt.savefig('fig2.png')
plt.show()
```

fig2.png:



Nasza funkcja ma taki wpływ na piksele obrazu. Sprawia, że piksele ciemniejsze stają się jeszcze ciemniejsze, a piksele jaśniejsze stają się jeszcze jaśniejsze. Prowadzi to do tego, że przy średnim kontraście kolory mogą być bardziej soczyste a obraz wyraźniejszy, lecz nie można z tym przesadzić, ponieważ w nadmiarze może to zdeformować kolory w niepożądanym kierunku.

b.

```
def transformacja_logarytmiczna(obraz):  1 usage
    obraz = obraz.copy()
    return obraz.point(lambda i: 255*np.log(1+i/255))
```

```
org=obraz
war1_1=transformacja_logarytmiczna(obraz)
war2_2=filtr_liniowy(obraz,  a: 2,  b: 100)
```

```
plt.figure(figsize=(16, 16))
plt.subplot( *args: 1,3,1) # ile obrazów w pionie, ile w poziomie, numer obrazu
plt.imshow(org)
plt.title("oryginalny obraz:")
plt.axis('off')
plt.subplot( *args: 1,3,2)
plt.title("wariant 1 - tranformacja logaritmyczna: ")
plt.imshow(war1_1)
plt.axis('off')
plt.subplot( *args: 1,3,3)
plt.title("wariant 2 - filtr liniowy a:2 b:100 ")
plt.imshow(war2_2)
plt.axis('off')
plt.subplots_adjust(wspace=0.05, hspace=0.05)
plt.savefig('fig3.png')
plt.show()
```

fig3.png:

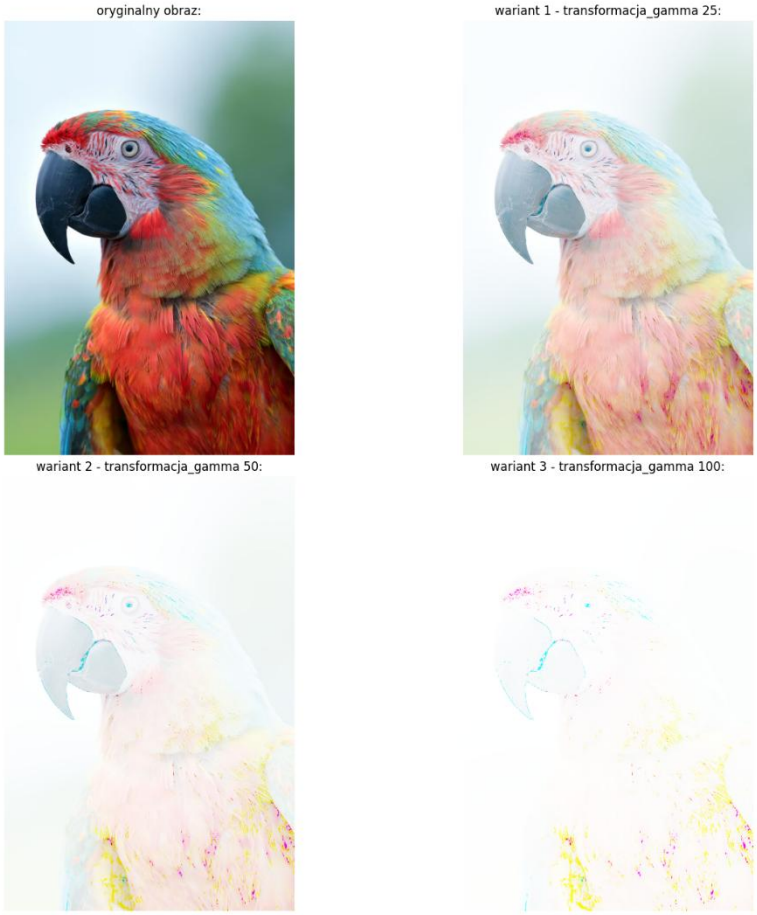
Obraz oryginalny jest oryginalny – niczym się nie różni. Obraz po zastosowaniu transformacji logarytmicznej pociemnił obraz i uwydatnił cienie, tam, gdzie były jasne elementy wciąż wydają się one jasne. Trzeci obrazek ekstremalnie pojaśnił obraz mnożąc jasność dwukrotnie i dodając jeszcze 100 jednostek do każdego kanału co jeszcze bardziej zwiększyło efekt jasności.

c.

```
def transformacja_gamma(obraz,gamma):  4 usages
    if gamma <= 0 or gamma > 100:
        return None
    obraz = obraz.copy()
    return obraz.point(lambda i: (i/255)**(1/gamma)*255)

org=obraz
war1=transformacja_gamma(obraz, gamma: 5)
war2=transformacja_gamma(obraz, gamma: 20)
war3=transformacja_gamma(obraz, gamma: 100)
```

fig4.png:



Wartość gamma wpływa na jasność obrazu, detale w cieniach zanikają, powoduje utratę kontrastu przy dużych wartościach.

5. Zadanie5

```python
def transformacja_gamma_lista(obraz, gamma):  1 usage
    if gamma <= 0 or gamma > 100:
        return None
    obraz = obraz.copy()
    lista =[]
    for i in range(256):
        wartosc = (i/255)**(1/gamma)*255
        lista.append(int(wartosc))
    lista = lista * 3
    return obraz.point(lista)

trans_gamma = transformacja_gamma(obraz, gamma: 0.5)
trans_gamma_lista = transformacja_gamma_lista(obraz, gamma: 0.5)
```

porównanie:

Funkcje działają identycznie.

6. Zadanie6
   a.

```
#a
T = np.array(obraz, dtype='uint8')
T += 100
obraz_wynik = Image.fromarray(T,  mode: "RGB")
obraz_wynik.show()

obrazz=obraz.copy()
obrazz.point(lambda i: i+100).show()
```

Z tego względu, że przy użyciu point (lambda i: i+100) obraz przykleja się do zakresu wartości 0 i 255 tzn., jeśli pixel ma wartość 320 po użyciu tej funkcji to wartość ta zamieni się w 255 a w przypadku Image.fromarray(T) obraz po prostu się zawinie przez to ze stanie się tam modulo 256.

   b.

```
def podobne_do_lambdy(obraz):  1 usage
    obraz = obraz.copy()
    obraz_t = np.array(obraz, dtype='uint8')
    w, h, d = obraz_t.shape
    for i in range(w):
        for j in range(h):
            r, g, b = obraz_t[i][j].astype(np.int_)
            r=r+100
            g=g+100
            b=b+100

            if r > 255: r = 255
            if g > 255: g = 255
            if b > 255: b = 255
            obraz_t[i][j][0] = r
            obraz_t[i][j][1] = g
            obraz_t[i][j][2] = b

    return Image.fromarray(obraz_t)

podobne_do_lambdy(obraz).show()
```