Raport lab 3-4 Mikołaj Małecki

1. Funkcja tworzy ramki grubości grub w odcieniach szarości zaczynając od czarnej, każda kolejna jest jaśniejsza o wartość zmiana_koloru

```
def rysuj_ramki_szare(w, h, grub, zmiana_koloru):
    t = (h, w)
    tab = np.zeros(t, dtype=np.uint8)
    ile = min(h, w) // grub
    for i in range(ile):
        tab[i*grub:h - i*grub, i*grub:w - i*grub] = (i*zmiana_koloru) % 256
    return Image.fromarray(tab)
```

Funkcja tworzy pasy pionowe grubości grub w odcieniach szarości zaczynając od czarnego, każdy kolejny jest jaśniejszy o wartość zmiana_koloru

```
def rysuj_pasy_pionowe_szare(w, h, grub, zmiana_koloru):
    t = (h, w)
    tab = np.ones(t, dtype=np.uint8)
    ile = int(w/grub)
    for k in range(ile):
        for g in range(grub):
            i = k * grub + g
            for j in range(h):
                tab[j, i] = (k*zmiana_koloru) % 256
    return Image.fromarray(tab)
```

2. Funkcja tworzy negatyw obrazu, jej sposób działania różni się w zależności od trybu obrazu. W przypadku trybu L i RGB odejmuje oryginalny odcień od wartości 255 aby uzyskać negatyw. Dla trybu 1 wykorzystywana jest funkcja negowania całej tablicy

```
def negatyw(obraz):
    tab = np.asarray(obraz)
    if len(tab.shape) == 2:
        h, w = tab.shape
    elif len(tab.shape) == 3:
        h, w, c = tab.shape
    if obraz.mode == 'L' or obraz.mode == 'RGB':
        tab_neg = tab.copy()
        for i in range(h):
            for j in range(w):
                tab_neg[i, j] = 255 - tab[i, j]
        return Image.fromarray(tab_neg.astype(np.uint8))
    elif obraz.mode == '1':
        tab_bool = np.asarray(obraz, dtype=bool)
        tab_neg = np.logical_not(tab_bool)
        return Image.fromarray(tab_neg)
    else:
        print('Neobsługwany typ obrazu')
        return obraz
```

2a) Obraz gwiazdka.bmp

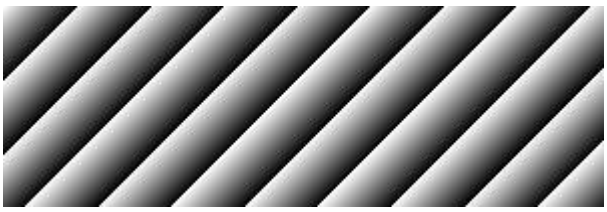
Obraz gwiazdka_negatyw.bmp

2b) Obraz rysuj_ramki_kolorowe(200, [20, 120, 220], 7, 7, -7) – ramki_kolorowe.png

Obraz ramki_kolorowe_negatyw.jpg

2c) Obraz rysuj_po_skosie_szare(100, 300, 7, 7) – skos_szare.png

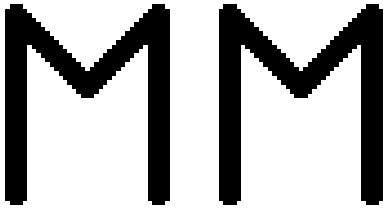
Obraz skos_szare_negatyw.png



3.

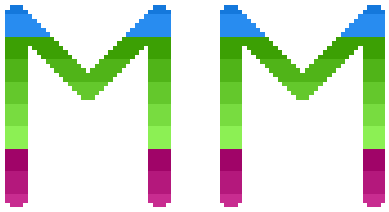
```
def koloruj_w_paski(obraz, grub, kolor, zmiana_koloru):
    t_obraz = np.asarray(obraz)
    h, w = t_obraz.shape
    tab = np.ones( shape: (h, w, 3), dtype=np.uint8) * 255
    ile = int(h / grub)
    for k in range(ile):
        r = (kolor[0] + k * zmiana_koloru) % 256
        g = (kolor[1] + k * zmiana_koloru) % 256
        b = (kolor[2] + k * zmiana_koloru) % 256
        kolory = [r, g, b]
        for m in range(grub):
            i = k * grub + m
            if i < h:
                for j in range(w):
                    if t_obraz[i, j] == 0:
                        tab[i, j] = kolory
    return Image.fromarray(tab)
```

3a) Obraz inicjaly.bmp



3b) Obraz koloruj_w_paski(inicjal, 5, [20, 120, 220], 20) – inicjaly_kolor.jpg

Obraz inicjaly_kolor.png



Obraz jpg na skutek kompresji stratnej jest bardziej rozmyty i występują na nim artefakty w przeciwieństwie do png stosującego kompresję bezstratną

4. W typie uint8 wartości przekształcane są przy użyciu modulo 265

a) dla 328 uzyskamy 72

b) dla -24 uzyskamy 232

5.

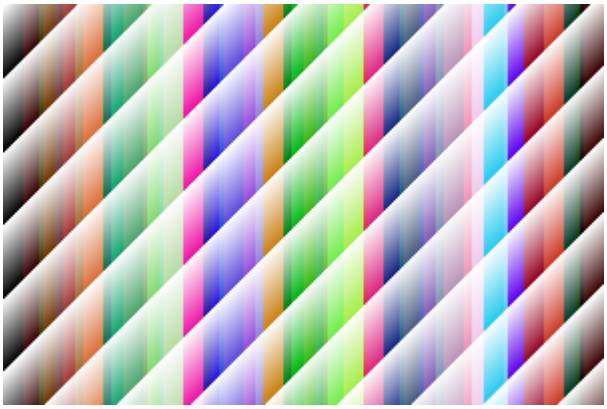
```python
def rysuj_pasy_pionowe_szare(w, h, grub, zmiana_koloru):
    t = (h, w)
    tab = np.ones(t, dtype=np.uint8)
    ile = int(w/grub)
    for k in range(ile):
        for g in range(grub):
            i = k * grub + g
            for j in range(h):
                tab[j, i] = (k*zmiana_koloru) % 256
    return Image.fromarray(tab)

obraz_r = rysuj_pasy_pionowe_szare( w: 300,  h: 200,  grub: 10,  zmiana_koloru: 55)
obraz_g = rysuj_pasy_pionowe_szare( w: 300,  h: 200,  grub: 18,  zmiana_koloru: 55)
obraz_b = rysuj_pasy_pionowe_szare( w: 300,  h: 200,  grub: 26,  zmiana_koloru: 55)

r_ext = np.expand_dims(obraz_r, axis=-1)
g_ext = np.expand_dims(obraz_g, axis=-1)
b_ext = np.expand_dims(obraz_b, axis=-1)

tab_rgb = np.concatenate( arrays: (r_ext, g_ext, b_ext), axis=-1)
obraz_rgb = Image.fromarray(tab_rgb, mode="RGB")
obraz_rgb.save("obraz6.png")
```

Obraz obraz6.png

6.

```python
def rysuj_po_skosie_szare(h, w, a, b):
    t = (h, w)
    tab = np.zeros(t, dtype=np.uint8)
    for i in range(h):
        for j in range(w):
            tab[i, j] = (a*i + b*j) % 256
    return Image.fromarray(tab)

obraz_a = rysuj_po_skosie_szare( h: 200, w: 300,  a: 7,  b: 7)

a_ext = np.expand_dims(obraz_a, axis=-1)

tab_rgba = np.concatenate( arrays: (obraz_rgb, a_ext), axis=-1)
obraz_rgba = Image.fromarray(tab_rgba)
obraz_rgba.save('obraz7.png')
```

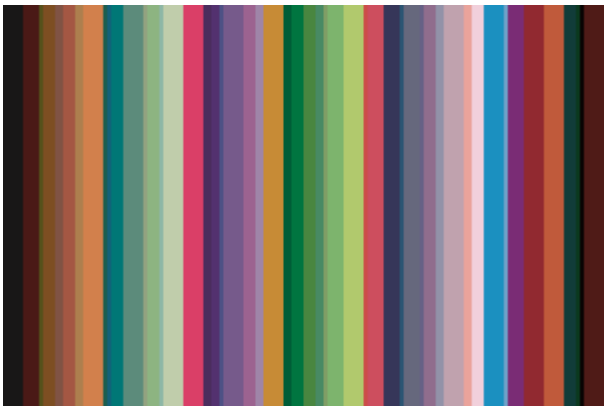
Obraz obraz7.png



7.

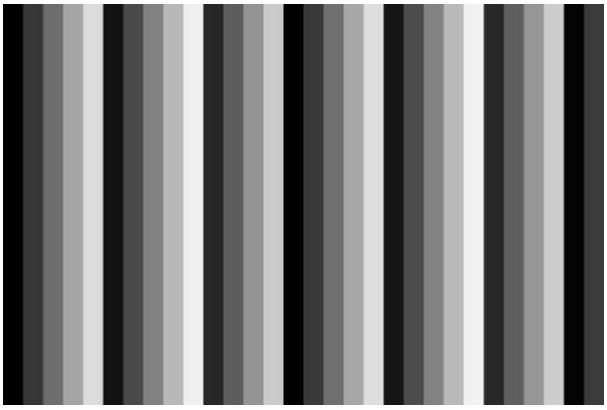
```python
def rgb_to_cmyk(rgb_array):
    rgb = rgb_array.astype(float) / 255
    r, g, b = rgb[..., 0], rgb[..., 1], rgb[..., 2]
    k = 1 - np.max(rgb, axis=2)
    c = (1 - r - k) / (1 - k + 1e-8)
    m = (1 - g - k) / (1 - k + 1e-8)
    y = (1 - b - k) / (1 - k + 1e-8)
    c[np.isnan(c)] = 0
    m[np.isnan(m)] = 0
    y[np.isnan(y)] = 0
    cmyk = np.stack( arrays: (c, m, y, k), axis=2) * 255
    return cmyk.astype(np.uint8)

tab_cmyk = rgb_to_cmyk(tab_rgb)
obraz_cmyk = Image.fromarray(tab_cmyk, mode="CMYK")
obraz_cmyk.save('obraz8.tiff')
```

Obraz obraz8.tiff

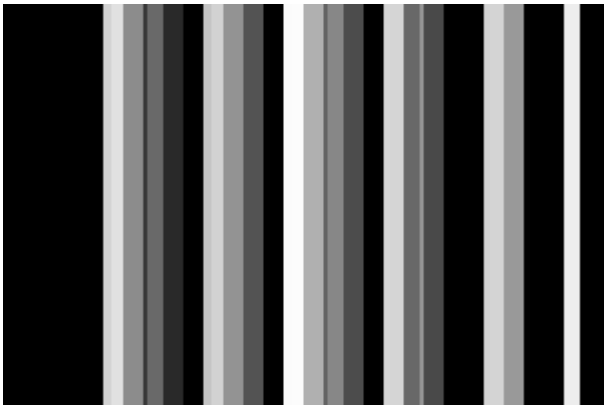


7a)

Obraz r.png



Obraz c.png



Obrazy różnią się grubością pasów, ponieważ r.png jest wynikiem funkcji rysuj_pasy_pionowe_szare, a c.png jest złożony z trzech wyników funkcji rysuj_pasy_pionowe_szare. Cyjan jest kolorem dopełniającym czerwonego, więc w miejscach w których czerwonego jest dużo (jasne paski na kanale r) cyjanu jest mało (ciemne paski)