

Zad 1

```
import random
import zlib
from PIL import Image

def generate_text_file(filename, length=100000):
    with open(filename, 'w') as f:
        f.write(''.join(random.choices('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789', k=length)))

def generate_high_compression_bmp(filename, width=1000, height=100):
    image = Image.new('RGB', (width, height), color=(255, 255, 255))
    image.save(filename)

def generate_low_compression_bmp(filename, width=1000, height=100):
    random_data = [random.randint(0, 255) for _ in range(width * height * 3)]
    image = Image.frombytes('RGB', (width, height), bytes(random_data))
    image.save(filename)

def compress_and_report(file_path):
    with open(file_path, 'rb') as f:
        compressed = zlib.compress(f.read())
    return len(compressed)

high_compression_bmp_filename = "high_compression.bmp"
low_compression_bmp_filename = "low_compression.bmp"

generate_text_file("random_text.txt")
generate_high_compression_bmp(high_compression_bmp_filename)
generate_low_compression_bmp(low_compression_bmp_filename)

compression_high = compress_and_report(high_compression_bmp_filename)
compression_low = compress_and_report(low_compression_bmp_filename)

print(f"BMP z największą kompresją rozmiar ZIP {compression_high} bajtów")
print(f"BMP z najmniejszą kompresją rozmiar ZIP {compression_low} bajtów")
```

random_text.txt

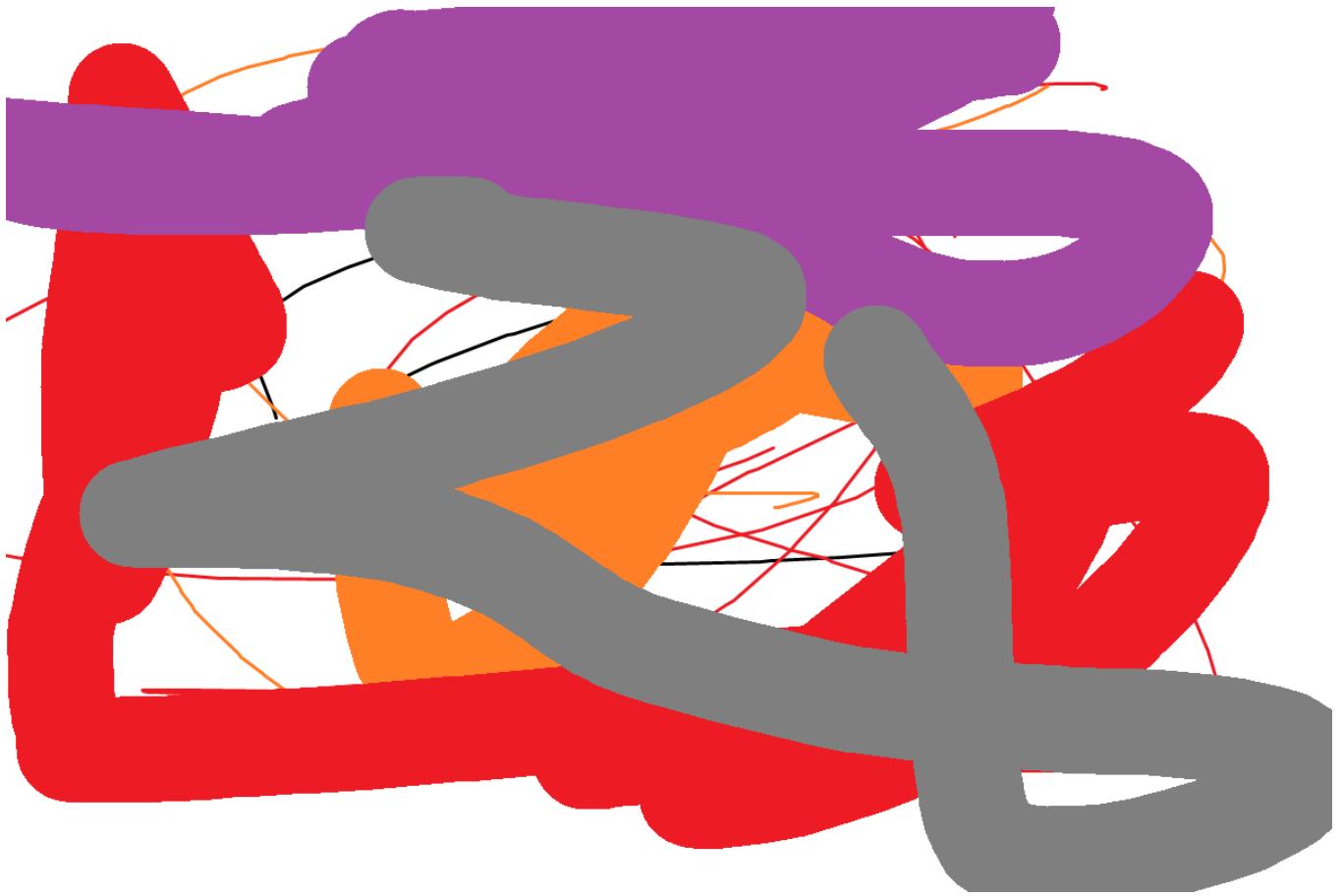
```
1 jEcQyGrIiHeNwJ9O5KsLoGeahibzq8WYx4C3Pn1Pt02UhnyjkdqB51wR82TUbY12qhoHmLT7817H4MtoL7815qoZ3pMB7J70lJeyHl9pcTxVxsryMoI6USL9Ax9F4B0qZeoj6oRo8qSmA2hdu8y10IH15  
kxxvD2SvfaiwgheeF04schX6yymtthaDqkjxjydv5weGmKfKg5Jsdznksv91BQ92gey09meCb5vzmdzu4klo0cUGoh1ap0ZpxePdmNSp2CFCis8hfioyXyzEgMYLrw23qoEj6n0tC1Xsmgrig  
iaQxP5pAzGTIMzK0tkyYznFicf1in5WfgtupHMuJgvVi1v08BeReeo2DQqjRhm2Kj1o4c4teQET5210usdMu122gMz6rF79m3Av3DRHyrcaFpg86Kea9oNwdMy31bc2H8D5rDzrbhpmbx1k13j  
Gdbf2zRtDxys0EiC1EV2pJGHYENJUK4tUlfJYV838478k1yvBncoMwP0y0vumWnRQ0ggy4avV1kZivAwACwrkmuuzgAVAm9lUr0jkumezzBfH8Rlw1wam0tD01BCEBdm1qXaGAUH09ry  
wi1aiBtKVtLH8FtVjZ1AtNDtLeuquqQn1t6pbbsjR7v1pt2pgKHfQTS1uQbBybrwao5pa5C1U83fGKRaLcn2Nrvg72r05Fqmf72wafXpQj1uam0dhX841cuw6X0Vtpv5h6r7c4jM0  
11YkhchB0Wk5LHgImysSY4gYLKx2f1DpvyQsMNTxIwgdl1Uf6Wmhd4R8qvhaq2vG45eK4z1Z7hJ6R871OxgrbzF8fT1SwYrnan17094z04BMpVzIMH1JA96c6W6cqgcXRxkfpypZbOn11Cz#W8Z7b1sda9  
jfYinEaqANDy1FsvFty1GhXxAXmZmpeEpInJ9WCZvBuNX1Y1laGworbL2KChHeyrQC9hIn5qZ62rP1ln24ks5Mzg3Pj1uSh1KyabQ5bJwMTHeCKFk1l0no3jCzRyOcdxSN4N73ehuF91SAy11YBsztU  
Bwkv1DyZ95g0a055Zj5NuEucBN9NQLRo9aF7h3jKkGWEUP5gHmzN2CnZ1HFfVz235CdF0Ay3YH6w0oDoF0Edx0XeeXrhpKm95zTz112p6y6pqtgbm2PxuonxzB37gvFdeukB00rgpYhiifb32Ruj1J0dML8ub  
NX04LUR6Kg3SC3pCXx1zF4rx5fghdQ3iA82P2APITN3yXnfiNBzarpA0MwTS6pFCOoA1VGKGzun6V1xPwH4wTasrF01CDxp18tgs7ubguA9pmrd5yBzXj1b7We58bam4051h6n3w1OKFD0hBwkmsuINzqx  
IVdviOZSV0xdY5xeagRaCmZ4P0C1b52Dm1k2ss56G0rnZM0384dK9Vj9femg9nHm8HJ0DTPHvUz8T4xRAaz8xm#HAD6WvVaYqDf8D77wale1j0MwZ5Km50Pg6hpgvDnpb3jY9o03msztw0N4ZwzT7NGV1g7Mq0  
gsik2wBa0IE5c1EV2pJGHYENJUK4tUlfJYV838478k1yvBncoMwP0y0vumWnRQ0ggy4avV1kZivAwACwrkmuuzgAVAm9lUr0jkumezzBfH8Rlw1wam0tD01BCEBdm1qXaGAUH09ry  
1f1TrdsoQu05F877rQn1yfM0mTeHq71kzHmkq56gmdku1d6a21lu1m0eq0ge3jKfz17k001jEz2zaPch3jjjJRHQwC1sp6386h66cqgcXRxkfpypZbOn11Cz#W8Z7b1sda9  
Y1F8kjpqdcmRCGB8LHgImysSY4gYLKx2f1DpvyQsMNTxIwgdl1Uf6Wmhd4R8qvhaq2vG45eK4z1Z7hJ6R871OxgrbzF8fT1SwYrnan17094z04BMpVzIMH1JA96c6W6cqgcXRxkfpypZbOn11Cz#W8Z7b1sda9  
9jedpxB17g377qmmUzgiVO6cBB1PPsx9HAWtVU7Rv7wyzTir1ndjh1K7T7xT5k83GvTy7DTeJesse1Xjrea6vB1f1nxehtD0QJuysblmmpf1fNkePp7EhtzqVfE3HBLNCfGggpuw2BhTT0ERHM  
v13Tk2zK5dt915Bw6uf237FB8uwF6Ggohhx2vUtriyY5WY1QfMKqMpDFesShKmzcevIw917X10qczkzRzkaUB8sldsewXzaci5udtmDyvb3uyzwf  
xNm9Yw3mCwfkguq00AA3n3b1v0Ou2oLUqaYnuSpwpx1jP5suez0S2Qe7EX1WkEevJBV2xQCV1mkCvku53ii1XNFSMaw03DT7q1gvzGr50hs1UuaBaTEdew948MDODDevLyleJdQl09MfksJ6Kr0tETNhAe  
mrA2cFlqUciftLw13PT13JzLvsB8s0dtK06vuJBmMK3AysNonTzY16y67ngzcsDRNMug9gbF77yKfz17sF0v5Bztd5d9k6A0E0qNLE0ffyL7y7Dzq1u19hC0MPk0ceX44PS8xRMmaehFzEIR13TnVt9iL  
JN10LFk40rsq21wYD0gtG1mWk1Ns56vhv1ktB14b5d0u8jYv5tqBz195m1w0z9M2PTMh1w1vCxxw8fPMovSw3w421j90tbo:v9jhmz2f1vRvdhu1zlkvFuwb  
MqKdDq74auUvYszL1w0x1ll1n72xAGUXv1V4tWohmtkdyOewGd07s7Pb6tvdq2dW67Tc2vldf4wuj2c6jzjyfikfqXASR10yf0zaxZkHmWn0ABvqah2GOImBaFrTdcjNpffxox8ZB7HGm41gQEBgjUt  
EgGt771DE77ap1tdauccs1Cbn015080Lijq8qz38jAZPzB1i1k8AgExQAxVTNjhaeJw1wB7DViyttaokv2P0F0QxtmNfcckrZhsDNg0jRV3hJ61Q8Fw1numXAUEKz5V7xpmQk1eluqak1ko2457dwgDCh  
t885Gf1pXx0lob1uaICONRUsdo8xtCuHuUvqo4ErhCa6c1R1SiuhZCGZDZwvtsodwCmE0b1A9Y8c2k8V1kA1NcxHRTRasR1N0BZG81wBwsqAfRnexxCxxatDmDt77t241kJKr1KwGxj84btu  
P9MmExCCTmU9fttIODEXKL93GahdUKAn574y2f0xCr27Ntup4FK1lDUw485BpCZUEm0Ah254v8UvUtohzCarryohByseRa3shUls3NPwVtsvTkttaGgq110PdVx02Dg3h1jN1NaCgH6GBo6vDA  
xc7nMutdgj1s5d6G21AFicRK6mbqP4G2erXt1pyAfmiNC86uSjnThAxeitAvX5q4679sXvgD6Dnjua8t83BLyjz7h7qBxaw44MuCNsai9Ne65jhsIX6Qle08fp7WmtSbKeFG179tjtjfaApFcs7u  
ovXn4KwPuvbJ1cY6G1hX0110fYyuwhgr1ws9rXqgallXu5K56wjs77rnLzm097H50Lzyrgzpt33K6gtCjgwy6XuMzxs0APCFg62cid0Lkmzd1lWzcpvx17f01PrFxk1BAkN9y8Ufus0g80KulMrrib  
pjzoCz4wHor4m5YRzLhHh1wB1jxROU3sSwugQj3KRT5tarHg172j0p3B8unaq5G9yGxkaJgxetvTaVfLGPakZLHvtH96HedoyLjaA1h9s4R1xWgI93iktx2n0jAjbkstBN19Yv2WRi10d1S3DjzYEVZ  
N517EwSbehihp2JFQVQcwHtphjuZxzxckBVy0g0zzsc2f16x50efzqCkyJutE6ygmFnyZ1vqMxe205TtQn5WQayVSeo2zRf1aCcc8EL0sazXde1l2w4X35ropXmbdyXRVG1MLBv3zE24M1ev22xd0622  
rLox9iuwCP3dd0YnIjNoz45sUj1b0RlwIHCnnUw1w79jdnRuagyRixhu3aprbn0UeR40cDmoXMGVNcx361uYp0f1EjGzD8jgAWyri1Df13zthYvCOPSM4Ef11d4rj1sMf8zPGqoRkRpkwqfBHayegKX6r  
t2ZGCPUbabsY20d4jcvXWbHr4ly77T766C4k1c1pk1xkh8VSc8Pj1Cmrg3DtonFv18R8CboH8t7tvCYQL8XgGNCNLHcp0EVao1CC7K0m2Tf0zq1ebvDAvRgoMk7mazh50t1ee05L6NbFbc1l95a9V1YnntQCl5xz79H00Hbf  
F0Fe6Ckq6euthHkOprrYljjnYwMWRfxK80D39Pz2XkF0zBm6oFZ0eG93j4yPVgnTU8374d71XpT4u0D2ffktw870ufCak3FEW013m56dcuCnT0CQo1mWQacmAxRLCCMa6zPQdK3CpvSdW006zpcqyHQ7q6k  
8G7Ma2kOj3jsj1C1e2b5xtd739MxuWBN3UVE5w0Qo1mB6sgdEQZVIPSxzj2ruh1hExwFtLeygkH753l1aNe3xG2zSpWm1z7sCnRInUgge7BzLs1v7uJ9-G24wxMw3jyjpxkZD51j9i3z08Y  
Q0mvnr7hdNDZCVb1hkuU8awM6U1f1Eu0fFhzgwu1nRQyLzEzIRAmgQdcocCh2k4e3Un0BfJ51M8r8qVx3ZhdQaR738vC0t8B0m0G6HjZbg2X5s09525j0hneCrUsfjVCEsdymg90kRs0pzeVV  
QcRvXejj3sVzHLfJ6eFMeB08qgwfRiindzYladomyC1T1m1CsInXNShh8gD6gexryq1kvyaPUKCeEsqyjkwgB41vyEHDz28TlxioZ0xnjBzZ2NNP0k8uwb6QtxRAdTQzC8w1Qd0wDVd90y96MtMupz2g  
p99K40uap5wOq491C1x3v089kv1a1ygaY1j1dRnk71s0Alpt0swmnl07yvdkh8Xcmzc7gHfsxvAy6x7s0Bvd1krmnZ3AchdGQvsPvfr70tfhiapF0xdsutoour52agt#M3C1p5999s9nszHx1ktvFhGo3cG5
```

high_compression.bmp



low_compression.bmp

Zad 2
Plik bmp:



```
from PIL import Image
import os

def convert_image_to_formats(input_file):
    image = Image.open(input_file)
    formats = ['PNG', 'GIF', 'JPEG']
    converted_files = {}

    for fmt in formats:
        output_file = f"{input_file.rsplit('.', 1)[0]}.{fmt.lower()}"
        image.save(output_file, fmt)
        converted_files[fmt] = output_file

    return converted_files

def calculate_file_size_and_compression_rate(original_file, converted_files):
    original_size = os.path.getsize(original_file)
    file_sizes = {'BMP': original_size}
    compression_rates = {}
    print(f"BMP: {original_size} bajtów")

    for fmt, file in converted_files.items():
        converted_size = os.path.getsize(file)
        file_sizes[fmt] = converted_size
        compression_rate = (original_size - converted_size) / original_size * 100
        compression_rates[fmt] = compression_rate
```

```
    print(f"\{fmt\}: {converted_size} bajtów, stopień kompresji:  
\{compression_rate:.2f}\%")  
  
    return file_sizes, compression_rates  
  
input_file = r"D:\Visual Studio Code\Codes\zad2.bmp"  
  
converted_files = convert_image_to_formats(input_file)  
  
file_sizes, compression_rates = calculate_file_size_and_compression_rate(input_file,  
converted_files)
```

BMP: 2880054 bajtów

PNG: 34706 bajtów, stopień kompresji: 98.79%

GIF: 20842 bajtów, stopień kompresji: 99.28%

JPEG: 57520 bajtów, stopień kompresji: 98.00%

Zad 3

$$H = \log_2 (n^2)$$

$$\log_2 (n^2) = 2\log_2 (n)$$

Entropia źródła wynosi $2\log_2 (n)$ bitów