Assignment-1 (Based on NumPy)

```
import numpy as np
from PIL import Image

# Q1: Questions on Basic NumPy Array

# (a) Reverse the NumPy array
arr = np.array([1, 2, 3, 6, 4, 5])
reversed_arr = arr[::-1]

# (b) Flatten the NumPy arr using two methods
array1 = np.array([[1, 2, 3], [2, 4, 5], [1, 2, 3]])
flatten1 = array1.flatten()
flatten2 = array1.ravel()

# (c) Compare numpy arrays
arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[1, 2], [3, 4]])
comparison = np.array_equal(arr1, arr2)

# (d) Most frequent value and indices
i = np.array([1,2,3,4,5,1,2,1,1,1])
y = np.array([1,1,1,2,3,4,2,4,3,3])
most_freq_i = np.bincount(i).argmax()
indices_i = np.where(i == most_freq_i)[0]
most_freq_y = np.bincount(y).argmax()
indices_y = np.where(y == most_freq_y)[0]

# (e) Sum operations
gfg = np.matrix('[4, 1, 9; 12, 3, 1; 4, 5, 6]')
all_sum = np.sum(gfg)
row_sum = np.sum(gfg, axis=1)
col_sum = np.sum(gfg, axis=0)

# (f) Matrix operations
n_array = np.array([[55, 25, 15],[30, 44, 2],[11, 45, 77]])
diag_sum = np.trace(n_array)
eigen_values, eigen_vectors = np.linalg.eig(n_array)
inverse_matrix = np.linalg.inv(n_array)
det_matrix = np.linalg.det(n_array)

# (g) Multiply and covariance
p1 = np.array([[1, 2], [2, 3]])
q1 = np.array([[4, 5], [6, 7]])
product1 = np.dot(p1, q1)
cov1 = np.cov(p1, q1)

p2 = np.array([[1, 2], [2, 3], [4, 5]])
q2 = np.array([[4, 5, 1], [6, 7, 2]])
product2 = np.dot(p2, q2)
```

```python
cov2 = np.cov(p2, q2)

# (h) Inner, outer and cartesian
x = np.array([[2, 3, 4], [3, 2, 9]])
y = np.array([[1, 5, 0], [5, 10, 3]])
inner_product = np.inner(x, y)
outer_product = np.outer(x, y)
cartesian_product = np.array(np.meshgrid(x.flatten(), y.flatten())).T.reshape(-1,2)

# Q2: NumPy Mathematics and Statistics

array = np.array([[1, -2, 3],[-4, 5, -6]])
abs_array = np.abs(array)
percentiles_flat = np.percentile(array, [25, 50, 75])
percentiles_col = np.percentile(array, [25, 50, 75], axis=0)
percentiles_row = np.percentile(array, [25, 50, 75], axis=1)

mean_flat = np.mean(array)
median_flat = np.median(array)
std_flat = np.std(array)

mean_col = np.mean(array, axis=0)
median_col = np.median(array, axis=0)
std_col = np.std(array, axis=0)

mean_row = np.mean(array, axis=1)
median_row = np.median(array, axis=1)
std_row = np.std(array, axis=1)

a = np.array([-1.8, -1.6, -0.5, 0.5,1.6, 1.8, 3.0])
floor_a = np.floor(a)
ceil_a = np.ceil(a)
trunc_a = np.trunc(a)
round_a = np.round(a)

# Q3: Searching and Sorting

arr3 = np.array([10, 52, 62, 16, 16, 54, 453])
sorted_arr = np.sort(arr3)
sorted_indices = np.argsort(arr3)
smallest4 = np.sort(arr3)[:4]
largest5 = np.sort(arr3)[-5:]

arr4 = np.array([1.0, 1.2, 2.2, 2.0, 3.0, 2.0])
integers_only = arr4[np.equal(np.mod(arr4, 1), 0)]
floats_only = arr4[np.not_equal(np.mod(arr4, 1), 0)]

# Q4: Image operations

def img_to_array(path):
    img = Image.open(path)
    img_array = np.array(img)
    if img.mode == 'RGB':
        np.savetxt('rgb_image.txt', img_array.reshape(-1, 3), fmt='%d')
    else:
        np.savetxt('grey_image.txt', img_array, fmt='%d')
    return img_array

# --------------------------------------------------
# Name: Aaditya Khanna
# Roll No: 102483002
# Sub Group: 3C53
# --------------------------------------------------
```