```
# question 2
import numpy as np
import matplotlib.pyplot as plt

in_file = np.loadtxt("in.dta.txt")
out_file =np.loadtxt("out.dta.txt")
transformed_in = np.empty([len(in_file), 8])
transformed_out = np.empty([len(out_file), 8])

for i in range(len(in_file)):
    x1 = in_file[i][0]
    x2 = in_file[i][1]
    transformed_in[i] = (1, x1, x2, np.power(x1, 2), np.power(x2, 2), x1 * x2,
np.abs(x1 - x2), np.abs(x1 + x2))


for i in range(len(out_file)):
    x1 = out_file[i][0]
    x2 = out_file[i][1]
    transformed_out[i] = (1, x1, x2, np.power(x1, 2), np.power(x2, 2), x1 * x2,
np.abs(x1 - x2), np.abs(x1 + x2))

in_error = 0
out_error = 0
processed = np.linalg.pinv(transformed_in)

weight = np.dot(processed, in_file)
newys = np.sign(np.dot(weight, np.transpose(in_file)))
check_out = np.sign(np.dot(weight, np.transpose(out_file)))

for i in range(len(newys)):
    if (newys[i][0] != in_file[i][2]):
        in_error += 1
for i in range(len(check_out)):
    if (check_out[i][0] != out_file[i][2]):
        out_error += 1
print float(in_error) / len(in_file)
print float(out_error) / len(out_file)
```

```
# questions 3-6
import numpy as np
import matplotlib.pyplot as plt

in_file = np.loadtxt("in.dta.txt")
out_file =np.loadtxt("out.dta.txt")
transformed_in = np.empty([len(in_file), 8])
transformed_out = np.empty([len(out_file), 8])

for i in range(len(in_file)):
    x1 = in_file[i][0]
    x2 = in_file[i][1]
    transformed_in[i] = (1, x1, x2, np.power(x1, 2), np.power(x2, 2), x1 * x2,
np.abs(x1 - x2), np.abs(x1 + x2))


for i in range(len(out_file)):
    x1 = out_file[i][0]
    x2 = out_file[i][1]
    transformed_out[i] = (1, x1, x2, np.power(x1, 2), np.power(x2, 2), x1 * x2,
np.abs(x1 - x2), np.abs(x1 + x2))

in_error = 0
out_error = 0
processed = np.linalg.pinv(transformed_in)

weight = np.dot(processed, in_file)
newys = np.sign(np.dot(weight, np.transpose(in_file)))
check_out = np.sign(np.dot(weight, np.transpose(out_file)))

for i in range(len(newys)):
    if (newys[i][0] != in_file[i][2]):
        in_error += 1
for i in range(len(check_out)):
    if (check_out[i][0] != out_file[i][2]):
        out_error += 1
k = -1
in_error += (np.power(10, k) / len(in_file)) * (np.sum(np.power(in_error, 2)))
print float(in_error) / len(in_file)
print float(out_error) / len(out_file)
```