```
%Problems 5-6

n = 0;
er = 1.0;
a = 0.1;
x = [1.0;1.0];

while er > 10^-14
    gradf = [2 * (x(1) * exp(x(2)) - 2*x(2)*exp(-x(1))) * (exp(x(2)) + 2 * x(2) *
exp(-x(1)));
        2 * (x(1) * exp(x(2)) - 2*x(2)*exp(-x(1))) * (x(1) * exp(x(2)) - 2 * exp(-
x(1)))];
    y = x - a*gradf;
    er = norm(y - x);
    x = y;
    n = n + 1;
end
n, x

%Problem 7

n = 0;
maxn = 15;
ers = 1.0;
a = 0.1;
xx = 1.0;
yy = 1.0;
er1 = 1.0;
er2 = 1.0;

while n < maxn
    gradf = [2 * (xx * exp(yy) - 2*yy*exp(-xx)) * (exp(yy) + 2 * yy * exp(-xx));
        2 * (xx * exp(yy) - 2*yy*exp(-xx)) * (xx * exp(yy) - 2 * exp(-xx))];
    y1 = xx - a*gradf(1);
    y2 = yy - a*gradf(2);
    er1 = y1 - xx;
    er2 = y2 - yy;
    er1 = norm(y1 - xx);
    er2 = norm(y2 - yy);
    ers = (er1 + er2) / 2;
    xx = y1;
    yy = y2;
    n = n + 1;
end
ers
```

```
#Problems 8-9

import numpy as np

runs = 100
d = 2

error = 0
points = 100
rate = 0.01
stop_param = 0.01
epochs = 0
for q in range(runs):
    weight = np.zeros(d + 1)
    new_weight = np.zeros(d + 1)
    xs = np.random.uniform(-1,1, (points, d))
    xs = np.c_[np.ones(points), xs]
    target_points = np.random.uniform(-1,1, (2, d))
    ys = np.zeros(points)
    for i in range(len(xs)):
        det = (xs[i][1] - target_points[0][0]) * (target_points[1][1] - \
            target_points[0][1]) - (xs[i][2] - target_points[0][1]) * \
            (target_points[1][0] - target_points[0][0])
        if det > 0:
            # on one side
            ys[i] = 1
        else:
            # on the other side
            ys[i] = -1
    while (True):
        epochs += 1
        np.random.shuffle(xs)
        for i in range(points):
            gradient = (ys[i] * xs[i]) / (1 + np.exp(ys[i] *
np.inner(weight,xs[i])))
            new_weight = new_weight + (gradient * rate)
        if (np.linalg.norm(weight - new_weight) < stop_param):
            break
        else:
            weight = new_weight

    for i in range(len(ys)):
        if (newys[i] != ys[i]):
            error += 1
print float(error) / (runs * points)
print float(epochs) / (runs)
```