```
# Tony Hyun Kim
# CS 224w, PS 3, Problem 4a
from __future__ import division # Non-truncating division
import snap

# Load the graph
#source = "gnm"
#source = "oregon1_010331"
source = "pa"

G  = snap.LoadEdgeList(snap.PUNGraph, source+".txt", 0, 1)
n0 = G.GetNodes() # Number of nodes in graph

# Deletion policy
scenario = 'f' # f: failure, a: attack

X = n0//100
Y = 50

# Output file
filename = source+"_X{}_Y{}_".format(X,Y)+scenario+".txt"
f = open(filename, 'w')

num_nodes_deleted = 0
while (num_nodes_deleted/n0 < Y/100):
    # Delete nodes in batches of X
    for _ in range(X):
        # Scenarios:
        #   In 'Attack's, delete nodes with maximum degree
        #   In 'Failure's, delete nodes at random
        if (scenario == 'a'):
            nid = snap.GetMxDegNId(G)
        else:
            nid = G.GetRndNId()
        G.DelNode(nid)
    num_nodes_deleted += X

    # Part a: Compute the diameter
    '''
    diam = snap.GetBfsFullDiam(G, 20)
    f.write("{0:.4f} {1}\n".format(num_nodes_deleted/n0, diam))
    '''

    # Part b: Fraction of nodes in the largest (strongly)
    #           connected component of a graph
    lcc_frac = snap.GetMxWccSz(G)
    f.write("{0:.4f} {1:.4f}\n".format(num_nodes_deleted/n0, lcc_frac))

print "Saved result to {}".format(filename)
f.close()
```