%Isaiah Griego

%Assignment 3

clear all;

close all;

clc;

%SEED

RandStream.setDefaultStream(RandStream('mt19937ar','seed',5021989));

%Phase I

%=======================================================================%

POP\_SIZE = 100;          %Population Size

PROB\_MUT = .998;         %Probability of mutation

PROB\_MULT\_MUT = .25;     %Probability of multiple mutations

MAX\_MUT = 4;             %The max number of multiple mutations

PROB\_CRO = 1;            %Probability of crossover

GENERATIONS = 200;       %Place holder to allocate inital memory

CHROME\_LENGTH = 24\*6;    %Length of the chromosomes must be a multiple of 6

BEST\_IND = zeros(1,114); %Holder for the best chromosome

GRAPH\_FRAMERATE = 4;     %How often the graph redraws itself in generations

INPUTS = zeros(256,8);

%This builds all the possible sortable 8-bit strings to determine whether

%the found sorting string is 'fit' meaning it can/can not sort the string.

for i = 0:255

    d2b = dec2bin(i,8);

    for j = 1:8

        INPUTS(i+1,j) = str2double(d2b(j));

    end

end

fitnesses = zeros(GENERATIONS,POP\_SIZE);

max\_fitnesses = zeros(1, GENERATIONS);

mean\_fitnesses = zeros(1, GENERATIONS);

gen = 0;

%This stops if and only if it has ran out of generations or we have found

%the solution that can sol

tic

while max(max\_fitnesses) < 256

    gen = gen + 1;

    %Generates initial test chromosomes only if it is the first

    %generation then proceeds to determine their fitness.

    if gen == 1

        chromosomes = round(rand(1, CHROME\_LENGTH, POP\_SIZE));

    end

    for i = 1:POP\_SIZE

        fitnesses(gen,i) = fitness(chromosomes(1,:,i), INPUTS);

    end

    max\_fitnesses(gen) = max(fitnesses(gen,:));

    mean\_fitnesses(gen) = mean(fitnesses(gen,:));

    %This determines whether or not we have a found a better individual

    %chromosome.

    if max\_fitnesses(gen)>fitness(BEST\_IND, INPUTS)

        BEST\_IND = chromosomes(1,:,find(fitnesses(gen,:)==max\_fitnesses(gen),1));

    end

    %Creates new chromosomes from the old chromosomes by mutation and

    %crossover.

    chromosomes = newchromes( chromosomes, fitnesses, gen,...

        PROB\_MUT, PROB\_CRO, BEST\_IND, PROB\_MULT\_MUT, MAX\_MUT );

    %Calculates how often to update the plot, will update the window every

    %GRAPH\_FRAMERATE frames

    if mod(gen,GRAPH\_FRAMERATE)==0

        plot(1:gen,(max\_fitnesses(1:gen)/256)\*100,1:gen,(mean\_fitnesses(1:gen)/256)\*100);

        xlabel('Generation');

        ylabel('Percentage of Corectly Sorted Strings');

        legend('Max', 'Mean','Location','Best');

        pause(1/480);

    end

end

plot(1:gen,(max\_fitnesses(1:gen)/256)\*100,1:gen,(mean\_fitnesses(1:gen)/256)\*100);

xlabel('Generation');

ylabel('Percentage of Corectly Sorted Strings');

legend('Max', 'Mean','Location','Best');

pause(1/480);

fprintf('Found the correct sorting algorithm in %d generations.\nWith %d compare exchanges, and a population size of %d.\n'...

    , gen,CHROME\_LENGTH/6, POP\_SIZE);

if max\_fitnesses(gen)>fitness(BEST\_IND, INPUTS)

    BEST\_IND = chromosomes(1,:,find(fitnesses(gen,:)==max\_fitnesses(gen),1));

end

fprintf('Winning Chromosome: \n%s\n', strrep(num2str(BEST\_IND), ' ', ''));

toc