CSE 420 Spring 2015 Program 1 Dr. Ronnie Ward

Due: March 2, 7pm.

The overall goal of this programming project is to successfully utilize the genetic algorithm (GA) to determine the best values for the "magic numbers" in Pandamat's code (see Lecture 4 slides).

Step 1 is to convert JeffsCode (initial code distributed for Homework 1) into a fitness function for use by GA. Do this by renaming main() to Jeffs_main(); add a class JeffsCode with Jeffs_main() as a member. Create a main() function that instantiates an instance of JeffsCode. The main() function should be able to call Jeffs_main().

Step 2 is to hack Jeffs_main() and the rest of the code to return an average number of problems solved in 1000 moves given a set of gene values (the gene values are from an individual in a GA population). Change Jeffs_main() to call iterate_pandemonium() a 1000 times (which is a 1000 moves) using the gene values passed for an individual. You will need to add code "here and there" in Jeff's Pandamat Code to track the number of problems solved and return it as a fitness value. I also advise you to "turn off" console output from Jeff's Code to help see the console output of your own code. Before going to the next step make sure you are convinced that a call to Jeff's_main() with a set of gene values (for example, pass it the original "magic constants" used in Jeff's code) properly calculates the number of problems solved in 1000 moves. If this is not correct, the rest of your efforts will be wasted.

Step 3 modify the main() function you added to implement the algorithm given in Figure 4.8. Input a random seed for use in creating random numbers. Create populations with 25 individuals. Use the allele examples given in slides in Lecture 4. An evolutionary step occurs each time a "new population" is created in Figure 4.8. We call this new population a generation. For a given seed, allow 20 generations to pass and then report the gene values and the fitness score for all 25 members in the final population. Compare results from runs with three (3) different seed values.

Issues:

a. How to represent an individual? You can use 5 real numbers (float in C++).
b. How to implement mating in the REPRODUCE(x,y) function in Figure 4.8?
c. How to enforce the range limits on the genes produced using the suggested allele values?

Questions:

A. How do the best gene values compare to the original "magic numbers"? If you plug these "best gene values" into the original pandamat code as new "magic constants", does the resulting code beat Pandamat? It should! Discuss why, or why not.
B. Outline the steps needed to evolve the demon connections in directly instead of using the learning method implemented in Pandament.

This assignment is worth 10 points. Young will tell you how the 10 points will be allocated and how your submission will be graded.