CSC 330 – Artificial Intelligence
Programming Project #4 – Genetic Algorithms
Due on Wednesday, April 3, by 5:00 PM

Description:
The Traveling Salesman problem (TSP) is a famous problem in Computer Science, both because it is a very hard problem to solve (one of a class called *NP-complete*), and because there are many approximation algorithms to find a decent solution to it. TSP can be quickly summarized by assuming that there are *N* cities, with a cost of travel from every city to every other city defined. TSP asks us to find the lowest cost path from a given city back to that city that passes through every other city once and only once. One way of obtaining a solution to a TSP problem is to use a genetic algorithm. You will be implementing a genetic algorithm to find a solution to TSP problems. This assignment will be more open-ended than assignments you may have previously had.

You may work on this project in pairs. You must inform me of who you are working with by next class period (Tuesday, March 19). Only one of the pair needs to tell me.

The Program:
When your program is run, it should do the following:
- Prompt the user to enter a file name.
- Open the file and read in the information about the TSP. If the file doesn't exist, you should print an appropriate error message and exit the program. If it does, print the information from the file (the list of names and grid of costs, see below).
- Run a genetic algorithm to find the best path for the TSP. **While running, your program should print the cost of the best tour it has found during each generation.**
- When your program is finished, print the best path found, and the resulting path cost.

You are free to store the TSP information any way you like, and to set up the genetic algorithm any way you like. It must be a genetic algorithm, though – using a population of candidate solutions, performing selection, cross-over, and mutation, etc. But the parameters and set-up are totally open. Your algorithm may not find the best solution (in fact, for larger problems it almost certainly will not), but it should "make progress" towards a solution.

File Format:
You can expect that the files that you are given for this problem will follow this format:
- The first line of the file is an integer N. This number is the number of cities in the problem.
- The next N lines will each contain one string, representing the name of the city. You can assume that this name will contain no spaces. The name on the i-th line is the name of the i-th city.
- The next N lines will each contain N integers. This will create an N x N grid which gives the costs to travel between cities. In particular, the number in the j-th column of the i-th row is the cost to travel from city i to city j. You cannot assume that the cost from i to j will be the same as the cost from j to i. The element in the i-th row and i-th column should be 0.

Here is an example of what a file like this might look like:

```
5
Arlington
Brazil
Chicago
Detroit
Evansville
0 3 1 4 2
4 0 6 1 4
1 2 0 6 5
6 3 1 0 8
4 1 9 3 0
```

There are some files on the Moodle assignment page for you to test your code on.

Specifications:

1. Your program represents a sincere attempt to solve the problem.
2. Your program compiles and runs, assuming specification 1 is met.
3. Your program successfully loads and prints information from the file entered by the user.
4. Your program prints the value of the best individual each generation.
5. Your program prints the best path and the best path cost at the end of the program.
6. Your program correctly implements the selection functionality of a genetic algorithm.
7. Your program correctly implements the cross-over functionality of a genetic algorithm.
8. Your program correctly implements the mutation functionality of a genetic algorithm.
9. Your program finds the optimal solution to the tsp.txt test file (6).
10. Your program finds the optimal solution to the tsp9.txt test file (332).
11. Your program makes good progress towards solving the tsp21.txt file.
12. Your program makes good progress towards solving the tsp99.txt file.
13. Your program makes good progress towards solving the indiana.txt file.
14. Your program makes good progress towards solving the us.txt file.
15. Your program finds the optimal solution to a small test file of my choosing.
16. Your program makes good progress towards solving a large test file of my choosing.
17. Your program is well-written and styled (formatting, consistent indentation, comments, etc.).