**Jacob Rook**

**Data Structure and Algorithms II**

**Project 3**

**User’s Manual**

**Setup and Compilation:**

1. Download and unzip the submission from eLearning on a Linux box in the multi-platform lab.
2. The includes:
   1. rook-j-p3.c
   2. TravelingSalesman.c
   3. TravelingSalesman.h
   4. makefile
   5. FunctionalDecomposition.docx
   6. UsersManual.docx
   7. cityWeights.txt
3. Environment: This program was tested on a combination of the Eclipse and jGrasp IDE. However, also works in the schools ssh server.
4. Compiling. This program includes a makefile. At the command line in Linux in the folder with the extracted files, type make into the command line. The program produces an executable: rook-j-p3

**Running the program:** Issue the command ./rook-j-p3. No command line arguments are required or checked.

User input: Program is user interactive. At the beginning of the program the program will ask the user for values needed for the traveling salesman program. After the user inputs all of the values the program will take over and run the simulation. Warning: there is no input validation, so any incorrect input will cause a run time error.

**Output:** All output goes to the console. Output will be similar to this:

Please enter the number of cities to run.

17

Please enter the number of individual tours are in a given generation

2000

Please enter the number of generation to run

1500

Please enter the percentage of generation that should be comprised of mutations.

50

Will a new cityWeights.txt be generated? (y/n)

N

Results:

Cities: 17

Optimal Coast from brute force: 338.037089

Time the brute force algorithm took: 157.560510

Cost from genetic algorithm: 389.479612

Time the genetic algorithm took: 103.810371

Percent of optimal the genetic algorithm produced: 115.21%