Functional Decomposition for the salesman problem:

main.c:

createMap(): this function creates a map based upon two txt files called theCities.txt and thePresets.txt. The program will exit if these two files are not found. It will return a map data structure.

findBrutePath(struct map*): This function generates the initial path based upon how many cities there are and then brute forces through all the possible paths extracting the shortest path from all possibilities. The best path is put into a best path array in the map structure. This function does not return anything.

printBestTrail(struct map*): This function prints out the best trail and it prints out the trails distance. It also returns the distance so that it can be tracked in main.

runMutation(struct map*): This function runs the mutation algorithm on the initial trail based upon the presets in thePresets.txt. It returns the location of the best trail in the array of trial arrays.

getDistanceOfTrailGeneration(struct map*, int location): This function calculates and returns the distance of the specified trail. If location is three then it calculates the location of the $4^{th}$ trail in the trail arrays.

printArrayLocation(struct map*, int location): This function prints out the array at the specified location in the trail array. For example if the location is 3 then it prints the $4^{th}$ location in the array of trials.

freeMap(struct map*: Frees all the memory used when creating the structure.

bruteSales.c:

swap(int* a, int b, int c): This function swaps two locations in the array a.

permuteIter(struct map*): This function creates the permutation numbers. It calculates the full running cycle of the brute force algorithm (n factorial) and the number of cities to run. It calls the permute function.

permute(struct map*, int nfact, int n): This function permutes all the possible iterations of the trails. Once each permutation is created then the path value is calculated. It is then compared with the best trail and if it Is better than the best is replaced and recorded.

findBrutePath(struct map*): This function calls the permuteIter function. This function is more aptly named and makes more sense for the user.

Map.c:

createMap(): see main.c

getDistance(struct map*, int* trail): This function calculates the trail distance of the given trail. It then returns the distance as a double.

getDistanceOfBestTrail(struct map*): This function calculates the distance of the trail in the best trail slot of the map structure. It returns the distance in the form of a double.

printPresets(struct map*): This prints out all the presets that were read by the program. This is for debugging purposes. No value it returned.

getNumCities(struct map*): This is a getter for the number of cities. It returns the number of citiies.

setBest(struct map, int* newBest): This function overwrite the best slot with the new array that is passed in through the newBest slot. Does not return anything.

getBestEntry(struct map*, int location): This is a getter to get the best location. Returns the best location.

printBestTrail(struct map*): This function prints the best trail and the distance of the best trail.

getDistanceOfTrailGeneration(struct map*, int start): This function get the distance of the passed in generation trail. The start integer is the column index of the trail to operate on.

calcArrayIndex(struct map*, int x, int y): Calcutes the effective 2d array location of the 1d array. It returns the corresponding 1d location.

printGeneration(struct map*): This prints out the entire generation array. Returns nothing.

printArrayLocation(struct map*, int start): This prints out the array starting at the passed in integer.

swapArray(int*a1, int*a2, int length): This function takes in the start of two arrays and their length. It then swaps the values of the arrays. This function does not return anything.

permuteOnce(struct map*, int start): This is a variant of the permute function that only goes through 1 iteration instead of n factorial iterations. For more information see the permute entry under the bruteSales c file.

premuteGenerations(struct map*, int start): Permutes all the generational arrays from start to end one time.

primeGenerations(struct map*): Primes all the generation entries with 1-n-1.

mutateArray(struct map*, int start): This function swaps the middle + 2 array entry with the middle – 2 array entry. Then it swaps the second entry with the last entry. Returns no value.

runMutation(struct map*): This function calls primeGeneration and the loops puts the best entries at the top of the array. Then it loops through numberOfGeneration times. During each cycle of the loop it permutes all but the very best trail and then find the best trails and relocates them to the top. The it mutates all the mutated arrays.

freeMap(struct map*): see main.c