Assignment 3 –

#include <iostream>

#include <vector>

#include <algorithm>

#include <cstdlib>

#include <ctime>

using namespace std;

// Type definition for representing a solution (chromosome) to TSP

using Chromosome = vector<int>;

// Function to calculate the cost of a tour in TSP

int calculateTourCost(const Chromosome& tour, const vector<vector<int>>& distanceMatrix) {

int cost = 0;

for (int i = 0; i < tour.size() - 1; ++i) {

cost += distanceMatrix[tour[i]][tour[i + 1]];

}

cost += distanceMatrix[tour.back()][tour[0]]; // Return to the starting city

return cost;

}

// Function to perform swap mutation on a chromosome representing TSP solution

void mutate(Chromosome& chromosome) {

int size = chromosome.size();

// Select two distinct positions randomly

int position1 = rand() % size;

int position2;

do {

position2 = rand() % size;

} while (position1 == position2);

// Swap the cities at the selected positions

swap(chromosome[position1], chromosome[position2]);

}

// Function to display a chromosome representing TSP solution

void displayChromosome(const Chromosome& chromosome) {

for (int city : chromosome) {

cout << city << " ";

}

cout << endl;

}

int main() {

// Sample TSP data (distance matrix)

vector<vector<int>> distanceMatrix = {

{0, 10, 15, 20},

{5, 0, 9, 10},

{6, 13, 0, 12},

{8, 8, 9, 0}

};

// Create an initial chromosome representing a solution to TSP

Chromosome chromosome = {0, 1, 2, 3};

cout << "Original Chromosome: ";

displayChromosome(chromosome);

// Calculate the cost of the original tour

int originalCost = calculateTourCost(chromosome, distanceMatrix);

cout << "Original Tour Cost: " << originalCost << endl;

// Perform mutation

mutate(chromosome);

cout << "Mutated Chromosome: ";

displayChromosome(chromosome);

// Calculate the cost of the mutated tour

int mutatedCost = calculateTourCost(chromosome, distanceMatrix);

cout << "Mutated Tour Cost: " << mutatedCost << endl;

return 0;

}