```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace _3P71_2
{
    class Program
    {
        //Data
        double[,] ulyssess22TSP =
        {
            {1,38.24,20.42},{2,39.57,26.15},{3,40.56,25.32},
            {4,36.26,23.12},{5,33.48,10.54},{6,37.56,12.19},
            {7,38.42,13.11},{8,37.52,20.44},{9,41.23,9.10},
            {10,41.17,13.05},{11,36.08,-5.21},{12,38.47,15.13},
            {13,38.15,15.35},{14,37.51,15.17},{15,35.49,14.32},
            {16,39.36,19.56},{17,38.09,24.36},{18,36.09,23.00},
            {19,40.44,13.57},{20,40.33,14.15},{21,40.37,14.23},
            {22,37.57,22.56}
        };

        double[,] eil51TSP =
        {
            {1,37,52},
            {2,49,49},
            {3,52,64},
            {4,20,26},
            {5,40,30},
            {6,21,47},
            {7,17,63},
            {8,31,62},
            {9,52,33},
            {10,51,21},
            {11,42,41},
            {12,31,32},
            {13,5,25},
            {14,12,42},
            {15,36,16},
            {16,52,41},
            {17,27,23},
            {18,17,33},
            {19,13,13},
            {20,57,58},
            {21,62,42},
            {22,42,57},
            {23,16,57},
            {24,8,52},
            {25,7,38},
            {26,27,68},
            {27,30,48},
            {28,43,67},
```

```
53              {29,58,48},
54              {30,58,27},
55              {31,37,69},
56              {32,38,46},
57              {33,46,10},
58              {34,61,33},
59              {35,62,63},
60              {36,63,69},
61              {37,32,22},
62              {38,45,35},
63              {39,59,15},
64              {40,5,6},
65              {41,10,17},
66              {42,21,10},
67              {43,5,64},
68              {44,30,15},
69              {45,39,10},
70              {46,32,39},
71              {47,25,32},
72              {48,25,55},
73              {49,48,28},
74              {50,56,37},
75              {51,30,40}
76          };
77
78          public enum CrossoverType
79          {
80              UOX,
81              PMX,
82              UOXPMX
83          }
84          public enum ElitismMode
85          {
86              top10P,
87              top1
88          }
89
90          public readonly string[] randomSeeds = new string[] {
91              "This is a real seed",
92              "Still a seed",
93              "*Slaps roof of car* You can fit so many seeds in this baby",
94              "I wonder if I'll loose marks over this",
95              "Yeet"
96          };
97
98          public Program()
99          {
100             //Setup
101             if (File.Exists("output.csv"))
102             {
103                 File.Delete("output.csv");
104             }
```

```
105             List<double[,]> datasets = new List<double[,]> { ulyssess22TSP,
                  eil51TSP };
106             string s = "";
107             do
108             {
109                 Console.Clear();
110                 Console.WriteLine("1. Output all permutations. 2. Output custom
                      run");
111                 s = Console.ReadLine();
112
113             }
114             while (s != "1" && s != "2");
115             switch (s)
116             {
117                 case "1":
118                     double[,] set = eil51TSP;
119                     int mPopSize = 1000;
120                     int mGenerationSpan = 1000;
121                     int tSize = 5;
122                     int round = 5;
123                     int startCity = 1;
124                     bool useConvergence = true;
125                     ElitismMode elitMode = ElitismMode.top10P;
126                     CrossoverType crossoverType = CrossoverType.PMX;
127                     double crossoverRate = 1.0;
128                     double mutateRate = 0.1;
129                     string seed = "SampleRandomSeed";
130                     City[] cit = new City[set.GetLength(0)];
131                     for (int i = 0; i < set.GetLength(0); i++)
132                     {
133                         cit[i] = new City(set[i, 1], set[i, 2]);
134                     }
135                     GeneticTS g1 = new GeneticTS(
136                         elitMode,
137                         crossoverType,
138                         crossoverRate,
139                         mutateRate,
140                         mPopSize,
141                         mGenerationSpan,
142                         seed.GetHashCode(),
143                         tSize,
144                         startCity,
145                         cit,
146                         useConvergence,
147                         round,
148                         CalculateSafeZone(elitMode, mPopSize)
149                     );
150                     Output1(g1, g1.bestFitnesses, g1.avgFitnesses, "CustomRun");
151                     break;
152                 case "2":
153                     List<double> crossovers = new List<double> { 1, .9 };
154                     List<double> mutations = new List<double> { 0, .1, 1 };
```

```
155
156                     int maxPopSize = 1000;
157                     int maxGenerationSpan = 1000;
158                     int tournamentSize = 5;
159                     int roundDigits = 5;
160                     int startCityIndex = 1;
161                     bool allowConvergence = true;
162                     foreach (double[,] dataset in datasets)
163                     {
164                         foreach (CrossoverType crossType in Enum.GetValues(typeof
                        (CrossoverType)))
165                         {
166                             foreach (double crossRate in crossovers)
167                             {
168                                 foreach (double mutRate in mutations)
169                                 {
170                                     foreach (ElitismMode eMode in Enum.GetValues
                                    (typeof(ElitismMode)))
171                                     {
172                                         GeneticTS geneticTS = null;
173                                         List<List<double>> setBest = new
                                        List<List<double>>();
174                                         List<List<double>> setAvg = new
                                        List<List<double>>();
175                                         string experimentName = string.Format
                                        ("SET: {0} Elietism: {1} Crossover Type: {2} Crossover Rate:
                                        {3} Mutation Rate: {4}",
176                                             dataset.GetLength(0), eMode,
                                            crossType, crossRate, mutRate);
177
178                                         Console.WriteLine("Starting " +
                                            experimentName);
179                                         for (int b = 0; b < randomSeeds.Length; b
                                            ++)
180                                         {
181                                             City[] cities = new City
                                            [dataset.GetLength(0)];
182                                             for (int i = 0; i < dataset.GetLength
                                            (0); i++)
183                                             {
184                                                 cities[i] = new City(dataset[i,
                                                1], dataset[i, 2]);
185                                             }
186                                             //Call GeneticTS
187                                             geneticTS = new GeneticTS(
188                                                 eMode,
189                                                 crossType,
190                                                 crossRate,
191                                                 mutRate,
192                                                 maxPopSize,
193                                                 maxGenerationSpan,
194                                                 randomSeeds[b].GetHashCode(),
```

```
195                                                 tournamentSize,
196                                                 startCityIndex,
197                                                 cities,
198                                                 allowConvergence,
199                                                 roundDigits,
200                                                 CalculateSafeZone(eMode,
                        maxPopSize)
201                                             );
202
203                                             setBest.Add(geneticTS.bestFitnesses);
204                                             setAvg.Add(geneticTS.avgFitnesses);
205                                         }
206                                         OutputAvg(geneticTS, setBest, setAvg,
                        experimentName);
207                                     }
208
209                                 }
210                             }
211                         }
212
213                     }
214                     break;
215             }
216         }
217         private int CalculateSafeZone(ElitismMode eMode, int maxPopSize)
218         {
219             switch (eMode)
220             {
221                 case ElitismMode.top10P:
222                     return (int)Math.Ceiling((double) maxPopSize * 0.1);
223                 case Program.ElitismMode.top1:
224                     return 1;
225                 default:
226                     return 0;
227             }
228         }
229         private void OutputAvg(GeneticTS geneticTS, List<List<double>>
              bestMaster, List<List<double>> avgMaster, string experimentName)
230         {
231             if (!File.Exists("output.csv"))
232             {
233                 File.Create("output.csv").Close();
234             }
235             StreamWriter data = new StreamWriter("output.csv", true);
236
237             string[] expInfo = geneticTS.GetExperimentInfo();
238             foreach (var line in expInfo)
239             {
240                 data.WriteLine(line);
241             }
242             data.WriteLine("");
243             data.WriteLine(experimentName);
```

```
244             data.WriteLine("Best Fitness " + "," + "Average Fitness");
245             //Depth
246             for (int i = 0; i < bestMaster.Max(x => x.Count) - 1; i++)
247             {
248                 //Breadth
249                 double avgBest = 0;
250                 double avgAvg = 0;
251                 for (int k = 0; k < bestMaster.Count; k++)
252                 {
253                     avgBest += (i >= bestMaster[k].Count) ? bestMaster[k].Last
                        () : bestMaster[k][i];
254                     avgAvg += (i >= avgMaster[k].Count) ? avgMaster[k].Last() :
                        avgMaster[k][i];
255                 }
256                 data.WriteLine(avgBest / bestMaster.Count + "," + avgAvg /
                      avgMaster.Count);
257             }
258             data.WriteLine("");
259             data.Close();
260             data.Dispose();
261         }
262         private void Output1(GeneticTS geneticTS, List<double> bestMaster,
              List<double> avgMaster, string experimentName)
263         {
264             if (!File.Exists("output.csv"))
265             {
266                 File.Create("output.csv").Close();
267             }
268             StreamWriter data = new StreamWriter("output.csv", true);
269 
270             string[] expInfo = geneticTS.GetExperimentInfo();
271             foreach (var line in expInfo)
272             {
273                 data.WriteLine(line);
274             }
275             data.WriteLine("");
276             data.WriteLine(experimentName);
277             data.WriteLine("Best Fitness " + "," + "Average Fitness");
278             for (int i = 0; i < bestMaster.Count; i++)
279             {
280                 data.WriteLine(bestMaster[i]+ ", " +  avgMaster[i]);
281             }
282             data.WriteLine("");
283             data.Close();
284             data.Dispose();
285         }
286         static void Main(string[] args) { Program P = new Program(); }
287     }
288 
289 }
290 
```