Name: Sander Rasmussen
Username: sanderas

Instructions on how to run program:

By running all the blocks in jupyter notebook, the default tests I did will run. If you want to change parameters you can do so in the bottom of each sections code blocks.

Exhaustive Search, change parameters in bottom section:

```
plan, altRoutes, time = exhaustiveSearch(6)
```

Hill Climbing, change parameters in bottom section:

```
path, length = bestHill(20,10) //where the first argument is the how many iterations the code will run, and the second is the amount of cities.
```

Genetic Algorithm, change parameters in bottom section:

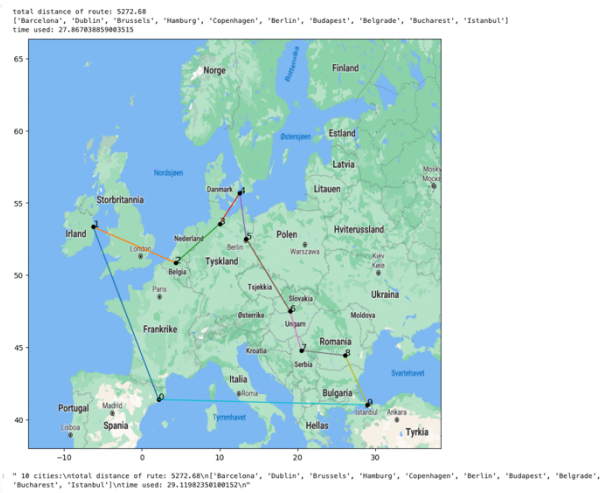
```
#6 cities
bestOf1, generationNumberX, avgBestGen1 = statisticsAndGA(24,8,generations,20)
#10 cities
bestOf2, generationNumberX, avgBestGen2 = statisticsAndGA(24,20,generations,20)
#24 citiess
bestOf3, generationNumberX, avgBestGen3 = statisticsAndGA(24,50,generations,20)

statisticsAndGA(24,50,generations,20) //where the first argument is amount of cities, the second is population, the third is how many generations, the last how many iterations.
```

Answers to questions in Assignment

Exhaustive Search

The algorithm is straight forward, it calculates the distance of every single permutation of the list containing all the cities and keeps the path which is the shortest.



The shortest tour among the first 10 cities is ['Barcelona', 'Dublin', 'Brussels', 'Hamburg', 'Copenhagen', 'Berlin', 'Budapest', 'Belgrade', 'Bucharest', 'Istanbul']. The distance was 5273 And the program used approximately 28 seconds to find it.

There are 133373577685028412444908147284377 6 permutations and it will take approximately 88726645 years to brute force TSP with 24 cities.

Hill Climbing

The way I implemented the hill climber was that I started out with a random permutation of the cities list, and then kept swapping to elements in the list. If the path was better than the best path recorded so far, it would be kept. If it was worse there was a chance it would be kept anyway in order to try to explore more. When the algorithm had failed at finding a better path a certain amount of times it would terminate and display the best it found.

Here are two runs, one with 10 cities and the other with 24.

Shortest path : 8268
Mean path : 11214
Worst path : 14763
Standard deviation : 1549
plot plan length : 8268
time : 0.021510805003345013

It performed acceptable when tested with 10 cities, but struggled more with 24 cities.

Shortest path : 24741

Mean path : 29697
Worst path : 34325
Standard deviation : 2608
plot plan length : 24741
time : 0.020717576997412834

Genetic Algorithm

I implemented it like they said in the lectures. I used partially Mapped Crossover to make offspring and swap mutation to mutate genes. It starts with a random permutation and then for every generation finds the fittest and keeps it. The parents are chosen in a roulette style, where the fitter the individual is, the higher the reproductive chance is.

```
average : 21604.15
worst : 24400.640000000003
mean : 21604.164000000004
standard deviation : 2014.0608640450512
Overall best distance : 16869.81
```

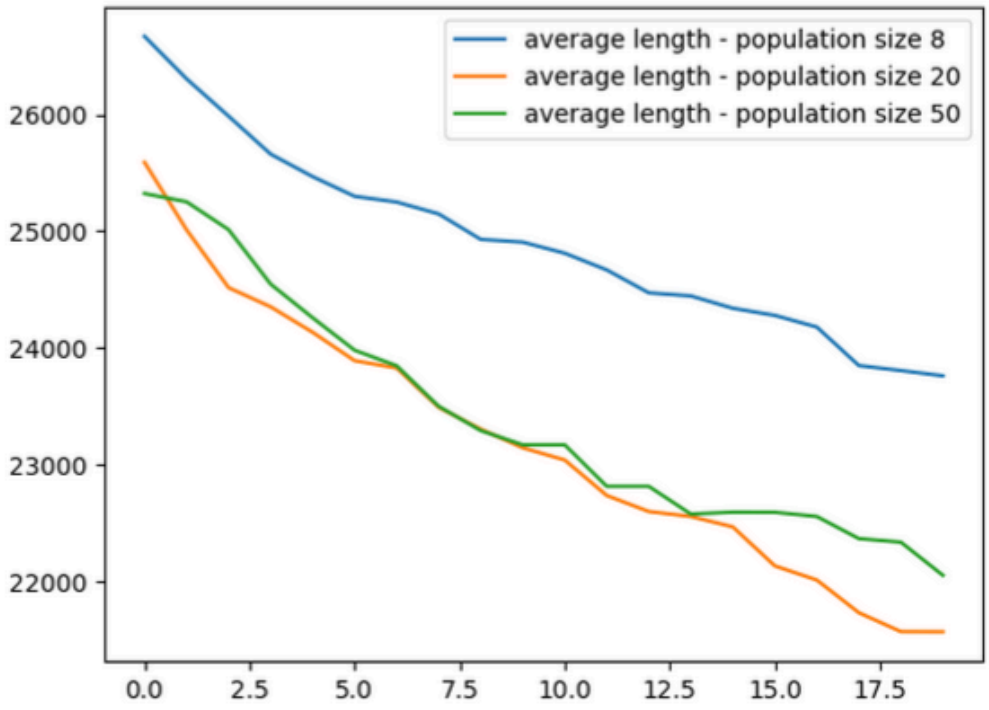
average : 21604.15
worst : 24400.640000000003
mean : 21604.164000000004
standard deviation : 2014.0608640450512
Overall best distance : 16869.81

```
[26670.3915, 26306.115, 25985.944, 25660.863, 25467.7065, 25298.624, 25249.5855, 25147.868000000002, 24929.5515, 24905.982500000002, 24811.
24667.8435, 24471.321, 24444.7085, 24339.0195, 24277.763, 24177.664, 23847.673, 23804.805, 23759.591500000002]
average : 19686.2
worst : 21306.84
mean : 19686.242499999997
standard deviation : 903.9812882138602
Overall best distance : 17779.78
```

average : 19686.2
worst : 21306.84
mean : 19686.242499999997
standard deviation : 903.9812882138602
Overall best distance : 17779.78

```
average : 19882.15
worst : 21787.600000000006
mean : 19882.175000000003
standard deviation : 1175.9773929598443
Overall best distance : 17303.359999999997
```

average : 19882.15
worst : 21787.600000000006
mean : 19882.175000000003
standard deviation : 1175.9773929598443
Overall best distance : 17303.359999999997



Although the best performance above was the GA with population size 50 (the biggest population size), it took a long time. I used 500 generations for demonstrational purposes , but the best overall result I got within a reasonable fast time was with a population size of 8 over 5000 generations. this gave me a better and
faster result than what I got from the GA with population 50 over 500 generations (see pictures below).

```
12376.56
['Moscow', 'Saint Petersburg', 'Stockholm', 'Prague', 'Vienna', 'Budapest', 'Belgrade', 'Sofia', 'Istanbul', 'Bucharest', 'Kyiv', 'Warsaw', '
Berlin', 'Copenhagen', 'Hamburg', 'Brussels', 'Paris', 'Munich', 'Milan', 'Rome', 'Barcelona', 'Madrid', 'London', 'Dublin']
```