In The Name of God

- Student Name: Melika Ahmadi Ranjbar
- Student ID: 521036
- Homework: #4 Genetic Report

Q1. In this question, we consider each city as the genes, and so we'll have the route as our chromosome.

As mentioned in the code using comments, we have 5 main steps we should go through.

We make the graph for city.

1.Creating initial population

2. Calculating fitness.

3. Selecting the best genes.

4. Crossing over.

5. Mutating to introduce variations.

6.Do them again.

So we first initialize some individuals, for our initial population, and we calculate their fitness values.

```
Population = []
InitialPopulation = 1000
UpperBound = 200
Iteration = 0
Mutation = 0.05
SizeOfSelection = 75

for i in range(InitialPopulation):
    Ind = Individual()
    Ind.Initialize()
    Ind.CalculateFitness()
    Population.append(Ind)
```

Individual class would be like:

```
def __init__(self):
    self.Chromosome = [None] * (CityNumber + 1)
    self.Fitness = 0
```

After that, we start the while of generations, we set a condition which here we're using a certain number of iterations (generations actually), and in this while, we first select parents with better fitness (here it means shorter path), and then we try to change them using crossover and with some probability doing mutation, we swap cities in between.

We again calculate the new fitness value and we'll finally have new population.

We have a certain number for choosing the best parents, and generating new population.

At the end we find the minimum value after all of the generations, and that is our answer (It might not be the optimal one, but it'll try to find one of the bests).

I've coded this question based on the Individual class, which has two property named: Chromosome and Fitness, and we also have two function, one of them initializes the route (chromosome indiscriminately), and the other one calculates its fitness value.

Another thing worth to explain might be the algorithm for opting best parents, which is just sorting the population and saving the minimums from that.

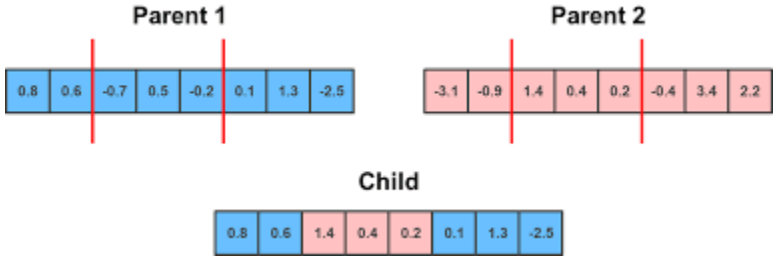
And because of the random thing, each time we might have different answers, look below:

```
[1, 3, 5, 7, 6, 4, 2, 1] Fitness:  63
```

```
[1, 2, 4, 6, 5, 3, 7, 1] Fitness:  64
```

The one and only point is, start city, we should choose the start city, it can also be done automatically but this way is more dependable.

I'm using two-point crossover which does the action below:



Q2. This question uses the exact same algorithm as the previous question, so we factor the steps and just explain the code and the differences with first question.

Here we have an equation and we want to find it's answer for 0, and there is only one answer for it which is in [-9, 9] range.

Again we initiate a population and start all of the steps mentioned for genetic algorithm.

Here we consider our chromosome, the actual float number for the equation and split it to an array, as you see here:

```
def __init__(self):
    self.Chromosome = [None] * ChromosomeSize
    self.Fitness = 0
    self.Answer = 0
    self.Neg = random.randint(0, 1)
```

Fitness is absolute value of equation using the answer we find.

Crossover and mutation and the while we're using is the same as before.

So we have y chromosome which keeps the Answer in array, like this:

```
[4, '.', 6, 3, 2, 1] 4.6321 Fitness:  0.07305144703786937
```