Sam Trenter
Project 1

Task1:

1.
Binary representation (over strings of size n) with a population of size N:
The class variable l stores that. Created on initialization.

2.
Random initialization of our population
CreatePopulation() in player.py

3.
Tournament selection (with configurable size tournaments) to determine parents
Tournament() in player.py

4.
Uniform Crossover with configurable probability to apply
UniformCrossover in Recombination.py

5.
Bit flip mutation (with probability of 1/n)
MutatePlayer() in player.py

6.
Elitism replacement
ElitismReplacement() in player.py

Task2:

(string size is the top column)

Run#	20	30	40	50	100	1000
1	7	10	14	25	41	521
2	5	12	17	22	40	526
3	7	7	17	19	44	427
4	4	13	19	15	35	460
5	8	13	12	21	38	437
AVG:	6.2	11	15.8	20.4	39.6	474.2

I’m guessing for 1,000,000 string size, it would have taken about 500,000 generations. Seeing that it only takes about half, I’m guessing the lower string size has a onemax time of less than half because you have a much higher chance of getting a “lucky string”, IE 75% 1s instead of an even 50/50 in the initial population.