Borna Tavassoli 810198374

Part 1. Each gene is a row in our 9×9 table, so a single chromosome has 9 genes.

Part 2. We build the chromosome and population in this part. About 1000 chromosomes should be enough to generate a solution.

Part 3. We generate a fitness function which returns the count of every number (from 1 to 9) in every row, column and 3×3 subtable. It's clear that the best fitness with this description is 243.

Part 4. We've defined the crossover and mutation functions here.

Part 5. Done!

Part 6. These are the following answers:

1. We choose about 70 percent of each population to build the new population from (via crossover or mutation operations). About 30 percent of the older population remains the same, but can eventually be cut if it's not efficient enough.
2. You can change the rate, but you need to remember to change the crossover and mutation rate respectively as well. All of these probability rates are co-related and you need to choose a set that is dynamic and efficient. It's a good practice to keep about 30 percent of the older generation in each turn because you have a little stability along with the possibility to generate new chromosomes!
3. We rely on these functions to generate the better chromosomes. Granted, we will generate a lot of inefficient chromosomes, but we need to pick a probability that will eventually generate out answer as well. If we pick a higher probability, we generate a bigger population every time so we will see our answer much sooner. On the other hand, if we pick a smaller probability, we might get stuck in a loop (because the demand to change the good answers is very low) and need to start over!
4. As mentioned above, if the chances to change the good answers are very low, we might get stuck in a loop, so we can define dynamic chance

variables that change in respect to the state of our program. If we see similar fitness values multiple times in a row, the chance to perform a crossover or mutation operation, increases. This will help lower the probability of getting stuck in a loop, however, we might need to start over the program in a more serious scenario.

Definitions:

1. Chromosome: A set of 9 genes (each gene represents a row in our sudoku table).
2. Population: About 1000 chromosomes, which we'll use to solve the puzzle.
3. Fitness Function: The summation of the number of times each number is repeated in each row, column and 3 × 3 table. The desired fitness value is 243.
4. Mutation Operation: We pick a chromosome and choose a row in it by random and swap two non-fixed cells in that row.
5. Crossover Operation: We choose two chromosomes and use a cyclic shift to change rows i to j in both of them. This means that two new chromosomes are generated which each will have rows i to j from one of the older chromosomes, and other rows, from the remaining older chromosome!

Execution time:

As mentioned, in the cases that we get stuck in a loop, we can't calculate the execution time. However, if we're lucky enough to set the probability variables properly (which can usually be achieved by setting both mutation and crossover chance to 1), the execution would take about 3-4 minutes. The final answers are copied into a file.