# Question 1:

import random  
  
POPULATION\_SIZE = 150  
GENES = '''1234567890,'''  
TARGET = ""  
  
  
class Individual(object):  
 def \_\_init\_\_(self, chromosome):  
 self.chromosome = chromosome  
 self.fitness = self.cal\_fitness()  
  
 @classmethod  
 def mutated\_genes(self):  
  
 global GENES  
 gene = random.choice(GENES)  
 return gene  
  
 @classmethod  
 def create\_gnome(self):  
  
 gnome\_len = 5  
 return [self.mutated\_genes() for \_ in range(gnome\_len)]  
  
 def mate(self, par2):  
 child\_chromosome = []  
 for gp1, gp2 in zip(self.chromosome, par2.chromosome):  
  
 prob = random.random()  
  
 if prob < 0.45:  
 child\_chromosome.append(gp1)  
  
 elif prob < 0.90:  
 child\_chromosome.append(gp2)  
  
 else:  
 child\_chromosome.append(self.mutated\_genes())  
  
 return Individual(child\_chromosome)  
  
 def cal\_fitness(self):  
 fitness = 0  
 for gs in self.chromosome:  
 if gs == ",":  
 continue  
 else:  
 s = int(gs)  
 if s % 2 == 0:  
 fitness += 1  
 return fitness  
  
  
def main():  
 global POPULATION\_SIZE  
  
 generation = 1  
  
 found = False  
 population = []  
  
 for \_ in range(POPULATION\_SIZE):  
 gnome = Individual.create\_gnome()  
 population.append(Individual(gnome))  
  
 while not found:  
  
 population = sorted(population, key=lambda x: x.fitness)  
  
 if population[0].fitness == 5:  
 found = True  
 break  
  
 new\_generation = []  
  
 s = int((10 \* POPULATION\_SIZE) / 100)  
 new\_generation.extend(population[:s])  
  
 s = int((90 \* POPULATION\_SIZE) / 100)  
 for \_ in range(s):  
 parent1 = random.choice(population[:50])  
 parent2 = random.choice(population[:50])  
 child = parent1.mate(parent2)  
 new\_generation.append(child)  
  
 population = new\_generation  
  
 print("Generation: {}\tString: {}\tFitness: {}". \  
 format(generation,  
 "".join(population[0].chromosome),  
 population[0].fitness))  
  
 generation += 1  
  
 print("Generation: {}\tString: {}\tFitness: {}". \  
 format(generation,  
 "".join(population[0].chromosome),  
 population[0].fitness))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

# Question 2:

* Chromosome 1: ABCDRGY
* Chromosome 2: ASTEYPI

AB | CDR | GY

AS | TEY | PI

Apply crossover:

Child 1: AB TEY GY = ABTEYGY

Child 2: AS CDR PI = ASCDRPI