Question 1 (a)

1. frontier = [('london', 0, ['london'])]
2. explored = {}
3. frontier = frontier - ('london', 0, ['london'])
4. explored = explored + london
5. frontier = frontier + ('birmingham', 110, ['london', 'birmingham'])
6. frontier = frontier + ('brighton', 52, ['london', 'brighton'])
7. frontier = frontier + ('bristol', 116, ['london', 'bristol'])
8. frontier = frontier + ('cambridge', 54, ['london', 'cambridge'])
9. frontier = frontier + ('cardiff', 161, ['london', 'cardiff'])
10. frontier = frontier + ('carlisle', 302, ['london', 'carlisle'])
11. frontier = frontier + ('dover', 71, ['london', 'dover'])
12. frontier = frontier + ('exeter', 172, ['london', 'exeter'])
13. frontier = frontier + ('glasgow', 396, ['london', 'glasgow'])
14. frontier = frontier + ('hull', 172, ['london', 'hull'])
15. frontier = frontier + ('leeds', 198, ['london', 'leeds'])
16. frontier = frontier + ('liverpool', 198, ['london', 'liverpool'])
17. frontier = frontier + ('oxford', 57, ['london', 'oxford'])
18. frontier = frontier - ('brighton', 52, ['london', 'brighton'])
19. explored = explored + brighton
20. frontier = frontier + ('aberystwyth', 301, ['london', 'brighton', 'aberystwyth'])
21. frontier = frontier + ('nottingham', 230, ['london', 'brighton', 'nottingham'])
22. frontier = frontier + ('penzance', 329, ['london', 'brighton', 'penzance'])
23. frontier = frontier + ('portsmouth', 101, ['london', 'brighton', 'portsmouth'])
24. frontier = frontier + ('sheffield', 268, ['london', 'brighton', 'sheffield'])
25. frontier = frontier + ('swansea', 288, ['london', 'brighton', 'swansea'])
26. frontier = frontier + ('york', 302, ['london', 'brighton', 'york'])
27. frontier = frontier - ('cambridge', 54, ['london', 'cambridge'])
28. explored = explored + cambridge
29. frontier = frontier - ('sheffield', 268, ['london', 'brighton', 'sheffield']) + ('sheffield', 170, ['london', 'cambridge', 'sheffield'])
30. frontier = frontier - ('swansea', 288, ['london', 'brighton', 'swansea']) + ('swansea', 270, ['london', 'cambridge', 'swansea'])
31. frontier = frontier - ('york', 302, ['london', 'brighton', 'york']) + ('york', 203, ['london', 'cambridge', 'york'])

Question 1 (b - 1)

Total cost: 502

Path: ['london', 'cambridge', 'york', 'aberdeen']

Question 1 (b - 2)

1. frontier = frontier - ('edinburgh', 387, ['london', 'cambridge', 'york', 'edinburgh'])
2. explored = explored + edinburgh

3. frontier = frontier - ('glasgow', 396, ['london', 'glasgow'])
4. explored = explored + glasgow
5. frontier = frontier - ('aberdeen', 502, ['london', 'cambridge', 'york', 'aberdeen'])

Question 1 (c)

driving_time (in hours) = Distance / velocity

air_pollution = 0.00001 * velocity per hour

Cost = driving_time + (0.00001 * velocity * driving_time)

$$= (\frac{distance}{velocity}) + (\frac{1}{100000} * velocity^2 * \frac{distance}{velocity})$$

$$= distance\left(\frac{1}{velocity} + \frac{velocity}{100000}\right)$$

The overall cost can be defined as the function given above. Therefore, to find the optimal solution, the cost function should be minimised. As distance is constant and cost id dependent on velocity, differentiating the above equation wrt velocity gives:

Gradient = $distance * \left(- velocity^{-2} + \frac{1}{100000}\right)$

Equating it to zero to find the minima, we get, velocity = 316.22 kmph

Therefore, the optimal path and the cost are given below:

Total cost: 3.1749267717664917

Path: ['london', 'cambridge', 'york', 'aberdeen']

Question 1 (d)

Intuitively, the best path will be driving at each road on its speed limit (if the speed limit is less than 300) as a fine of 1000 increases the cost exponentially and the path no longer remains the optimal path. Thus, the optimal path of the solution will be:

Solution: ('aberdeen', 300.0, ['london', 'liverpool', 'edinburgh', 'aberdeen'])

['london', 'liverpool', 'edinburgh', 'aberdeen']

Question 2 (a)

CHUKN088IS and _9CE_TW1CK

Question 2 (b)

The password is defined by a class "Chromosome" with sequence being the password string and fitness being the fitness value assigned by blurry_memory method. When initialising the population, password string is generated randomly from the allowed chars (uppercase string, numbers 0-9 and _). Population is defined as the list of Chromosome. A tournament policy is used for selection. In this policy, "n" random individuals are selected and the top two individuals with the highest fitness are chosen for crossover. An N-point crossover has been defined. The genes (sequence or password string) are crossed over at "n" random points. The two resulting children are then added to the population with their fitness=None. A Mutation probability is defined which governs the chances of mutation in a chromosome of a new population. Mutation happens of random gene (password string) in a sequence by changing the char at the point with a randomly generated char.

Question 2 (c)

The statistics given below are based on these hyper-parameters:

POPULATION_SIZE = 100 # population size of each generation

MUTATION_PROB = 0.05 # probability of mutation in a generation

TOURNAMENT_SIZE = 3 # number of participation for tournament selection

CROSSOVER_POINTS = 2 # N for n-point crossover

For PASSWORD_TYPE = 0 or 1 # the password type (0 or 1)

As number of reproduction = (Population_size/2) * (number_of_generation – 1) (since, one crossover results in two individuals and the initial population is not a result of reproduction)

Sr. no.	No. of Generation (P_type = 0)	No. of reproduction (P_type = 0)	No. of Generation (P_type = 1)	No. of reproduction (P_type = 1)
1	111	5500	151	7500
2	231	11500	123	6100
3	328	16350	262	13050
4	150	7450	253	12600
5	218	10850	260	12950
6	519	25900	155	7700
7	167	8300	90	4450
8	302	15050	225	11200
9	229	11400	278	13850
10	195	9700	152	7550
Average (mean)	245	12200	194.9	9695
Variance	13551.11111	33877777.78	4606.766667	11516916.67
Standard deviation	116.4092398	5820.46199	67.87316603	3393.658301

The mean, variance and standard deviation are shown in the table above for 10 random runs of the genetic algorithm for both the password 0 and 1. It can be seen that the algorithm take about 200 generations at a population size of 100 and a reproduction of 11000 before finding the answer.

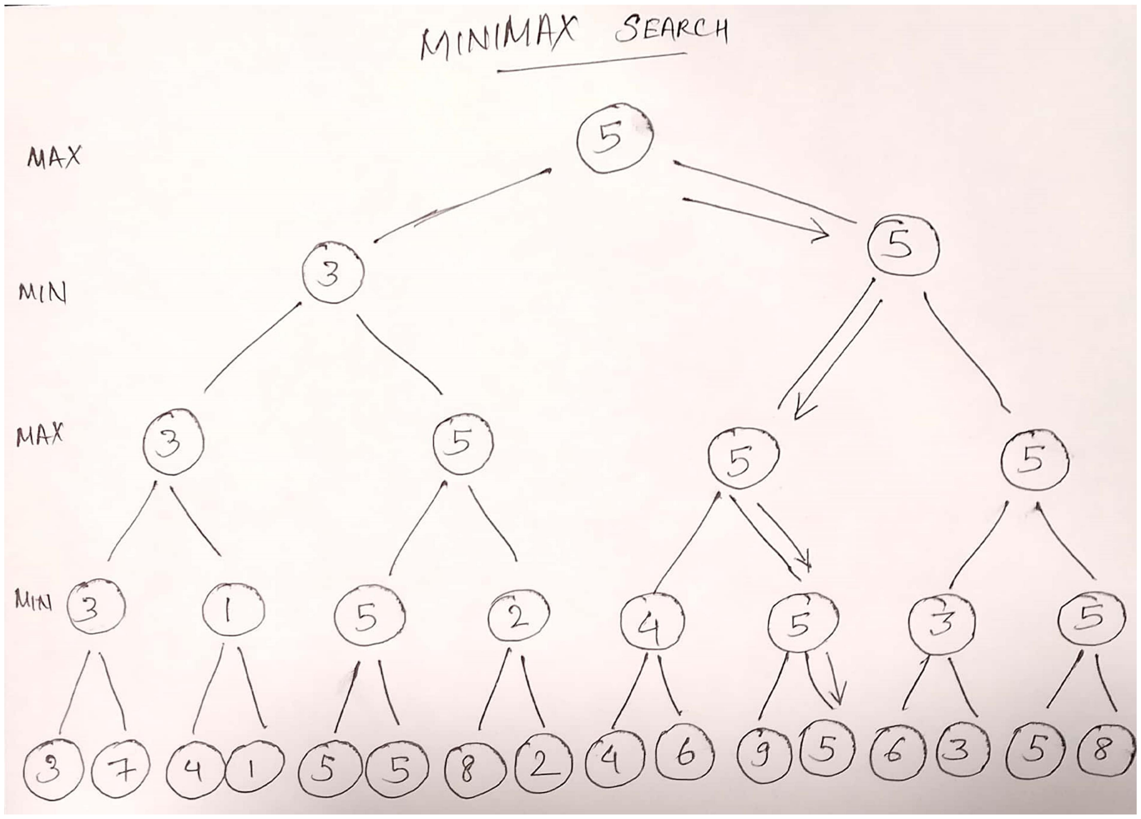
Question 2 (d)

The results obtained below were at random.seed(5) with MUTATION_PROB = 0.05 , TOURNAMENT_SIZE = 3, and CROSSOVER_POINTS = 2

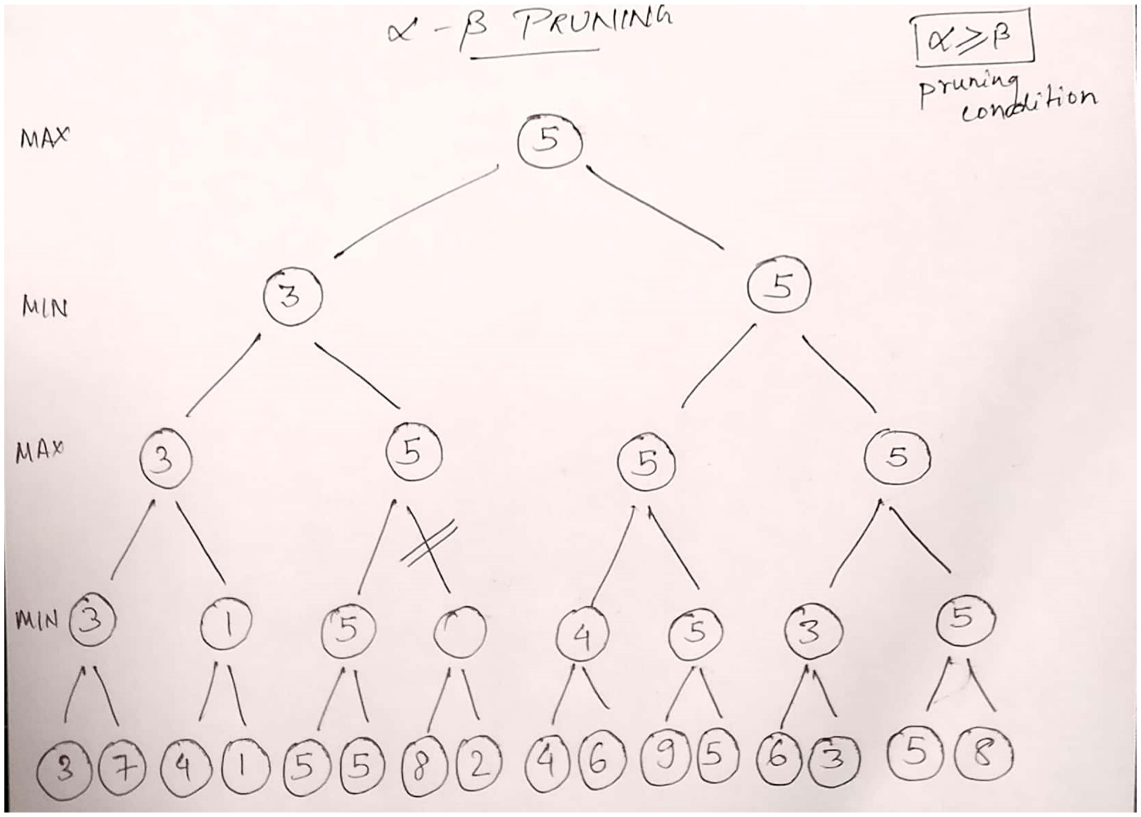
Sr. No	Password type	Population Size	No. of Generation
1	0	10	4279
2	1	10	5416
3	0	100	143
4	1	100	118
5	0	1000	22
6	1	1000	20
7	0	10000	20
8	1	10000	20

A smaller population size (10) results in more generation before finding the solutions. By increasing the population size (100 or 1000), the number of generations reduces significantly. But after a certain point, the increase in the population size (10000) stops showing any significant reduction in the number of generations produced. Also, with a larger population size, the search time increases.

Question 3 (a)



Question 3 (b)



The given node is pruned because, the MAX player (at depth 3) will choose a node which is greater than or equal to 5. The MIN player (at depth 2) will always choose 3 as it is lesser than 5 (or any number greater or equal to 5). Therefore, exploring the right child of the MAX node (at depth 3) is not worth as the game will never go down that path (cause the MIN player will play optimally).

Question 3 (c - 1)

As the terminal state of the game is 5. Being the MAX player, the game is worth playing when x ranges from 0 to 4 (MAX player will get a profit). At x = 5, MAX player will break even with zero profit and zero loss. At x > 5, the MAX player will be in loss.

Question 3 (c – 2)

It is profitable to be the MAX player if the value of x < 5. With x > 5 it is better to be the MIN player.

As the MAX player has to pay x units to start the game and the MIN player has to pay MAX the amount equal to the terminal state of the tree (in this case 5). Therefore, MAX earns (5 – x) units whereas, MIN earns (x – 5) units. So for x < 5, MIN player is in loss whereas MAX player is in profit and vice versa for x > 5. At x = 5, there is no profit or loss.