Question 1:

a) Following the given pseudo-code of the UCS, manually execute it for three iterations. Initial state,

```
1. frontier = [('london', 0, ['london'])]
2. explored = {}
```

1st iteration, all of the neighbors of London will be inserted into the frontier and sort them with the path cost, meanwhile record the only city which has been explored ('london').

```
1. frontier = [('brighton', 52, ['london', 'brighton']), ('cambridge', 54, ['london',
   'cambridge']), ('oxford', 57, ['london', 'oxford']), ('dover', 71, ['london', 'dove
   r']), ('birmingham', 110, ['london', 'birmingham']), ('bristol', 116, ['london', 'b
   ristol']), ('cardiff', 161, ['london', 'cardiff']), ('exeter', 172, ['london', 'exe
   ter']), ('hull', 172, ['london', 'hull']), ('leeds', 198, ['london', 'leeds']), ('l
   iverpool', 198, ['london', 'liverpool']), ('carlisle', 302, ['london', 'carlisle'])
   , ('glasgow', 396, ['london', 'glasgow'])]
2. explored = {'london'}
```

2nd iteration, since the distance from London to Brighton is the lowest (52), so we will give priority to explore the city – Brighton (using the 'pop' function). Next, we explore the neighbors of Brighton unexplored. And record the second city we explored ('brighton')

```
1. frontier = [('cambridge', 54, ['london', 'cambridge']), ('oxford', 57, ['london', '
   oxford']), ('dover', 71, ['london', 'dover']), ('portsmouth', 101, ['london', 'brig
   hton', 'portsmouth']), ('birmingham', 110, ['london', 'birmingham']), ('bristol', 1
   16, ['london', 'bristol']), ('cardiff', 161, ['london', 'cardiff']), ('exeter', 172
   , ['london', 'exeter']), ('hull', 172, ['london', 'hull']), ('leeds', 198, ['london
   ', 'leeds']), ('liverpool', 198, ['london', 'liverpool']), ('nottingham', 230, ['lo
   ndon', 'brighton', 'nottingham']), ('sheffield', 268, ['london', 'brighton', 'sheff
   ield']), ('swansea', 288, ['london', 'brighton', 'swansea']), ('aberystwyth', 301,
   ['london', 'brighton', 'aberystwyth']), ('carlisle', 302, ['london', 'carlisle']),
   ('york', 302, ['london', 'brighton', 'york']), ('penzance', 329, ['london', 'bright
   on', 'penzance']), ('glasgow', 396, ['london', 'glasgow'])]
2. explored = {'london', 'brighton'}
```

3rd iteration, now the first node in the frontier has the lowest path cost (54), so we decide to explore the neighbors of the city – Cambridge, and it will be recorded into explored set.

```
1. frontier = [('oxford', 57, ['london', 'oxford']), ('dover', 71, ['london', 'dover']
   ), ('portsmouth', 101, ['london', 'brighton', 'portsmouth']), ('birmingham', 110, [
   'london', 'birmingham']), ('bristol', 116, ['london', 'bristol']), ('cardiff', 161,
    ['london', 'cardiff']), ('sheffield', 170, ['london', 'cambridge', 'sheffield']),
   ('exeter', 172, ['london', 'exeter']), ('hull', 172, ['london', 'hull']), ('leeds',
    198, ['london', 'leeds']), ('liverpool', 198, ['london', 'liverpool']), ('york', 2
   03, ['london', 'cambridge', 'york']), ('nottingham', 230, ['london', 'brighton', 'n
   ottingham']), ('swansea', 270, ['london', 'cambridge', 'swansea']), ('aberystwyth',
    301, ['london', 'brighton', 'aberystwyth']), ('carlisle', 302, ['london', 'carlisl
   e']), ('penzance', 329, ['london', 'brighton', 'penzance']), ('glasgow', 396, ['lon
   don', 'glasgow'])]
2. explored = {'london', 'brighton', 'cambridge'}
```

b) b-1.

The shortest path from London to Aberdeen is shown below and the corresponding total cost (i.e. the total driving distance) is 502.

```
['london', 'cambridge', 'york', 'aberdeen']
```

b-2.

The last two iterations are listed below.

```
1. frontier = [('glasgow', 396, ['london', 'glasgow']), ('aberdeen', 502, ['london', '
   cambridge', 'york', 'aberdeen'])]
2. explored = {'brighton', 'bristol', 'cardiff', 'leeds', 'aberystwyth', 'exeter', 'ca
   mbridge', 'nottingham', 'birmingham', 'carlisle', 'manchester', 'swansea', 'oxford'
   , 'liverpool', 'hull', 'sheffield', 'london', 'portsmouth', 'edinburgh', 'york', 'd
   over', 'newcastle', 'penzance'}
3. frontier = [('aberdeen', 502, ['london', 'cambridge', 'york', 'aberdeen'])]
4. explored = {'brighton', 'bristol', 'cardiff', 'leeds', 'aberystwyth', 'exeter', 'ca
   mbridge', 'nottingham', 'birmingham', 'carlisle', 'manchester', 'swansea', 'oxford'
   , 'liverpool', 'hull', 'sheffield', 'london', 'portsmouth', 'glasgow', 'edinburgh',
    'york', 'dover', 'newcastle', 'penzance'}
```

c) For the cost between any two neighboring cities, it can be notated the formula below.

$$cost_i = cost_P + cost_T = \frac{S_i}{v_i} \times 0.00001 \times {v_i}^2 + \frac{S_i}{v_i} \quad (1)$$

$cost_P$ and $cost_T$ denote the pollution and time cost between the two neigboring cities respectively. S_i represents the distance (km) between the two cities and v_i denotes the driving speed (km/h).

Let the derivation of formula (1) equals zero, i.e. $\frac{\partial cost_i}{\partial v_i} = 0$, we can get $v_i \approx 316$. In other words, the total cost between any two neighboring cities is the lowest when the driving speed is 316 (km/h).

According to (b), the shorest path distance is 502 km from London to Aberdeen, therefore the lowest overall cost roughly equals 3.17.

$$cost_{shorest} = \left(0.00001 \times v_i + \frac{1}{v_i}\right) \times S_{shorest} \approx 3.17$$

d) For each road between any two neighboring cities, the cost can be represented as the formula below. The cost on each road is comprised of the rental fee and the likely fine.

$$cost_i = cost_r + cost_f = \frac{d_i}{v_i} \cdot rent + p \cdot fine \quad (2)$$

On each road, the rental cost is the time you spend on the road multiplied by the rental fee, and the fine cost is the probability p of you will be fined multiplied by the fine. Meanwhile, the probability p is related to the driving speed v_i.

In order to find the lowest cost on each road, since the maximum speed of the supercar is 300 km/h, I use the brute force searching (experiment each speed within the range from 1 to 300 km/h).

```
['london', 'birmingham', 'york', 'aberdeen']
300.0
```

The best path is shown above, and the overall cost is 300 units.

Question 2:

a) 'R1DD7E4U__' and '7ROLL_F4CE' are the two passwords I got. (My student ID is 190189237).

b) The state encoding is represented by a 10 characters-long string. Selection is to select the elites in the current population for producing the offspring of the next generation. The next step (crossover) is to use such selected parents for mating in order to generate the offspring. In mutation, each individual has a probability to mutate (some genes are replaced by random genes) to maintain diversity and avoid premature convergence.

c) Run my code 100 times, the average number of reproductions to converge the first password is 92.35 and the corresponding variance is 1255.17. And for the second password the average number of reproduction and variance is 82.48 and 1185.25 respectively.

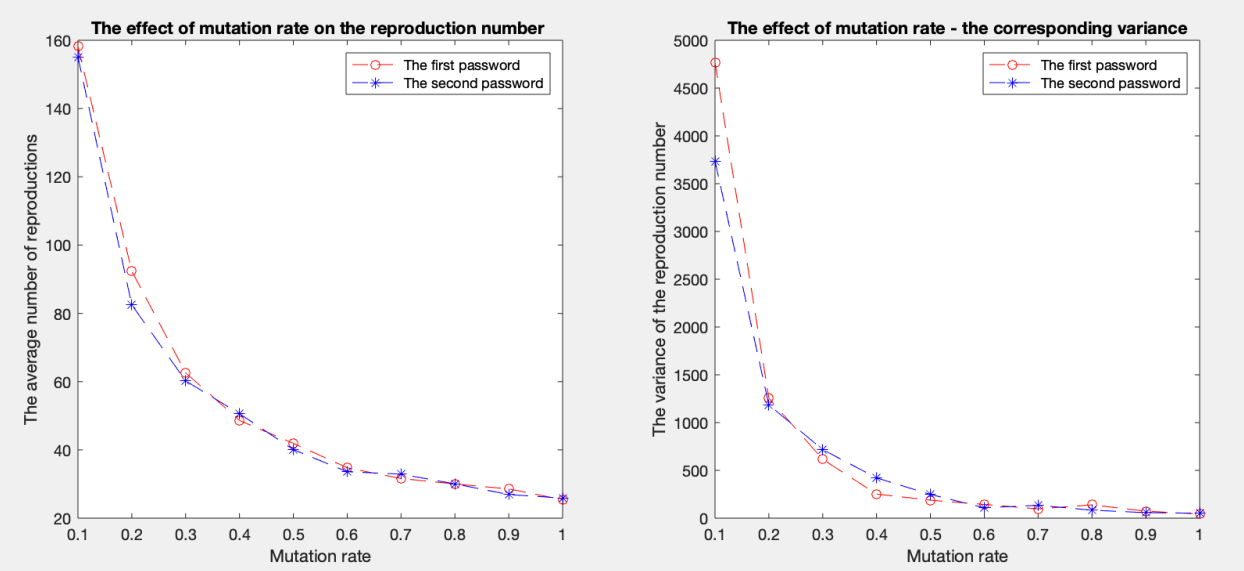
```
Run the code 100 times.
The averaged number of reproductions (the first password): 92.35
The variance of reproduction number (the first password) : 1255.1675000000002
The averaged number of reproductions (the second password): 82.48
The variance of reproduction number (the second password) : 1185.2496
```

d) The parameters of my code are shown below:

```
# Parameters
geneSet = get_gene_set()  # The element set of password: {'A' - 'Z', '0' - '9', '_'}
pwd_length = 10  # The length of the password
mutate_rate = 0.2  # The probability of mutation
population_size = 80  # The number of the initial population i.e. some random passwords are created to evolute
number_best_candidates = 20  # The number of best candidates which are selected from the population
number_offspring = 20  # For the best candidates, the number of their offspring
```

I decide to discuss the effect of the hyper-parameters (mutation rate) on the performance of my algorithm.

I change the hyper-parameter (mutation rate) when keeping the others unchanged. All parameters are preset in the figure above.



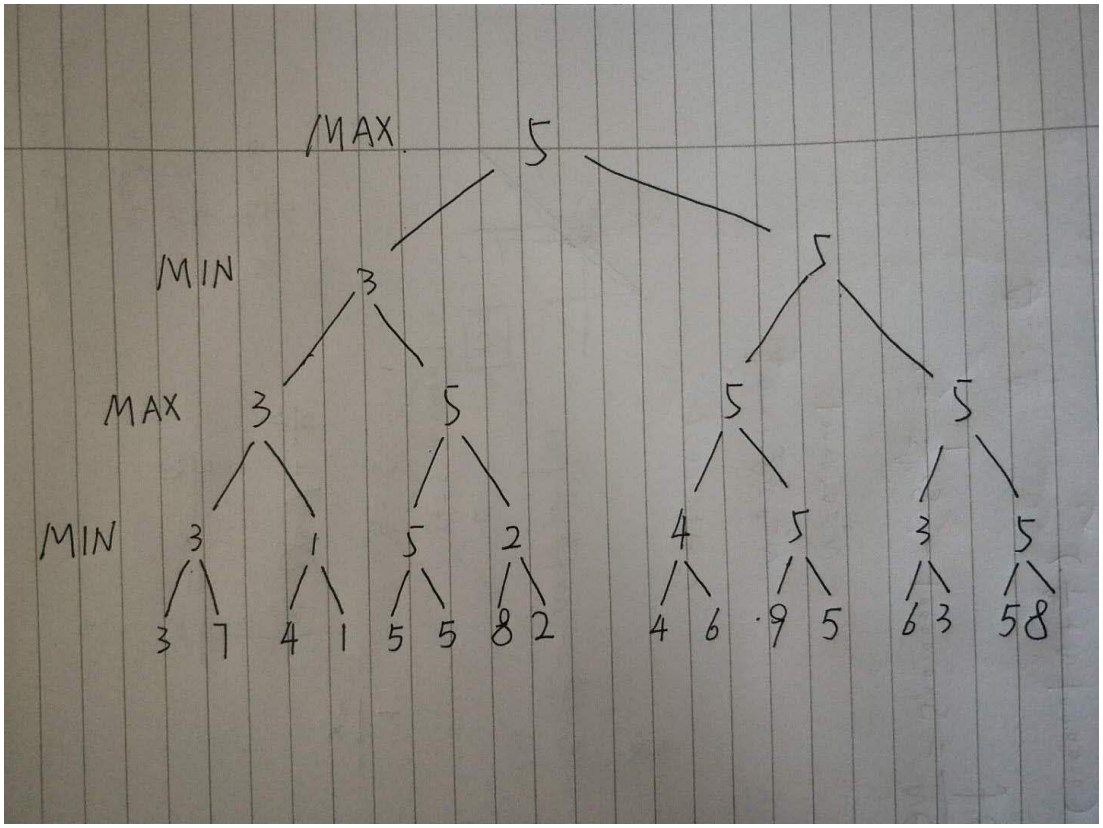
From the left image, it is clear that the required number of reproductions dramatically decrease with the increase of mutation rate for both of the two passwords. In other

words, the higher the mutation rate is, the better the algorithm performs. In addition, the reproduction number slowly stabilize and reach to roughly 30 after the mutation rate is greater than 0.7.

From the right image, we can find that a low mutation rate will result in a high variance of the corresponding reproduction number, i.e. the lower the mutation rate is, the more unstable the number of reproductions is.

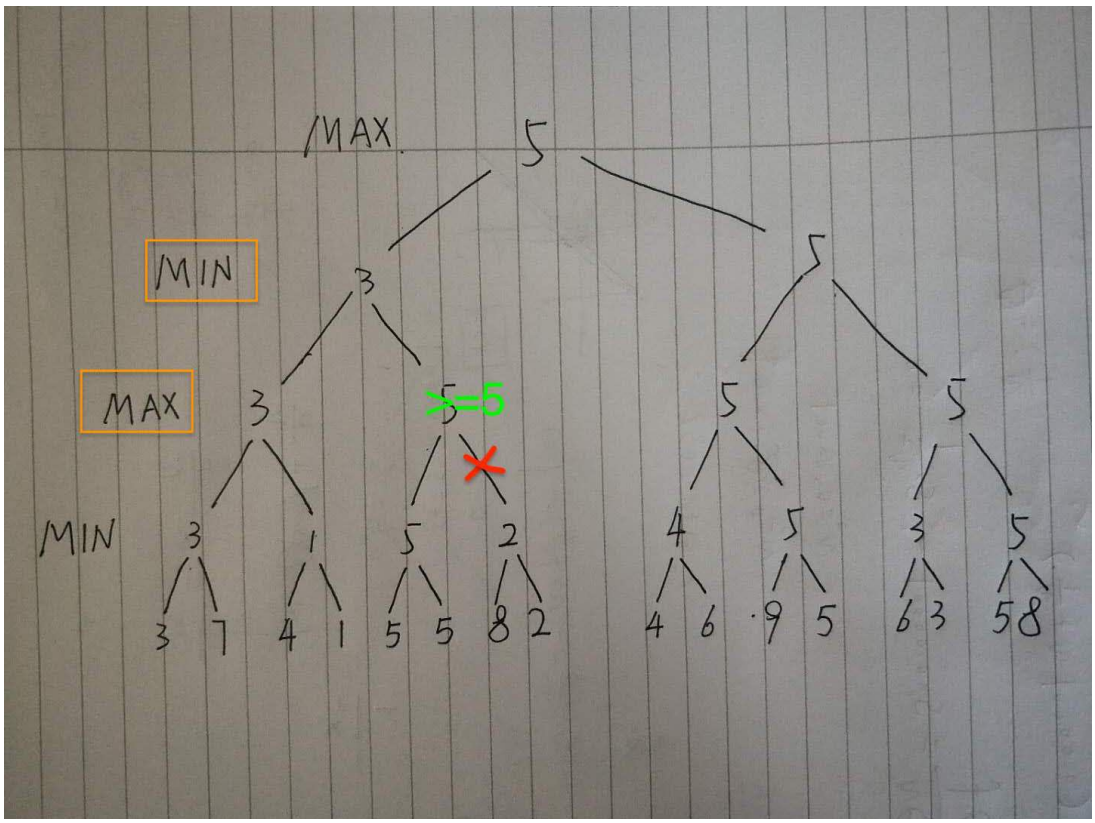
Question 3:

a)



b) As shown below, the red cross is where I prune, and it is a beta pruning.

In the green position, there is a lower limit on the achievable score. For the MAX player, the green node should be greater than or equal to 5, since by exploring the early branches the possible value equals to 5. For the MIN player will always prefer 3 than a value ≥ 5.



c) c-1

The range of the value x is between 0 and 9, i.e. [0,9), that will be worth playing the game for the MAX player. This is because, due to the terminal states of Figure 2 given, the reward that the MAX player wins ranges from 1 to 9, which means the entry fee x should be lower than the reward.

c-2

When the fixed value of x is low (such as x <= 4), I prefer to be the first (MAX) player, because there is a relatively high probability that I can earn money. When the fixed value of x is high (such as 4<x<9), I prefer to be the second (MIN) player, because there is a relatively high probability that I can earn money.