IERG3050_2021 Spring_Assignment 4

Name: To Yi Him SID: 1155126366

Part A,B,C

All outputs are shown in the folder.

Config.csv can modify the input.

Most of the explanation of program code are commented inside the program: assignment4_basic.py

All functions required work properly, and most of the output are similar to the sample result under same seed.

There are still some printed data in the console for debugging, you can ignore them

Here are some output showing:

Output of config_seed0.csv

D	100		
N	100		
T_size	12		
T_detail	[2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 4]		
p	0.8		
k	2		
Customer_	customer_seed0.csv		
Result_file	simulation_result.csv		
Summary_	simulation_summary.csv		

Output of customer_seed0.csv

	A	B	C	D	E	F	G	H
1	customer_	dining_tin	carrier	arrival_time_day_1	arrival_time_day_2	arrival_time_day_3	arrival_tin	arrival_ti
2	1	45	1	12:10	-1	-1	-1	-1
3	2	55	0	11:34	12:56	11:35	-1	-1
4	3	48	0	12:59	-1	-1	-1	-1
5	4	45	0	12:24	12:08	11:59	11:34	11:10
6	5	41	0	12:42	12:09	11:13	11:36	12:01
7	6	50	0	12:49	-1	-1	-1	-1
8	7	41	0	12:37	12:35	-1	-1	-1
9	8	74	0	12:18	-1	-1	-1	-1
10	9	96	0	12:12	12:24	-1	-1	-1
11	10	39	0	11:55	-1	-1	-1	-1
12	11	61	1	12:10	-1	-1	-1	-1
13	12	45	0	11:38	-1	-1	-1	-1
14	13	46	0	12:35	12:43	11:42	11:17	-1
15	14	81	0	11:15	-1	-1	-1	-1
16	15	31	0	12:56	12:38	11:39	12:40	11:15
17	16	31	0	11:40	11:26	11:49	-1	-1
18	17	30	0	12:10	-1	-1	-1	-1
19	18	65	0	11:53	-1	-1	-1	-1
20	19	60	0	12:43	-1	-1	-1	-1
21	20	70	0	12:36	12:50	12:10	-1	-1
22	21	106	1	11:24	-1	-1	-1	-1
23	22	62	0	11:04	12:52	-1	-1	-1
24	23	42	0	12:22	12:09	-1	-1	-1

Output the result of the simulation: simulation_result.csv or simulation_result_v1.csv

A	B	C	D	E	F	G
Day	Time	Event	len(Queue	Number_c	Number_finish	
1	11:02	Customer	0	0	0	
1	11:04	Customer	0	1	0	
1	11:05	Customer	0	2	0	
1	11:06	Customer	0	3	0	
1	11:07	Customer	0	4	0	
1	11:07	Customer	1	4	0	
1	11:09	Customer	0	6	0	
1	11:11	Customer	0	7	0	
1	11:12	Customer	0	8	0	
1	11:12	Customer	1	8	0	
1	11:13	Customer	0	10	0	
1	11:13	Customer	1	10	0	
1	11:14	Customer	0	12	0	
1	11:15	Customer	0	13	0	
1	11:15	Customer	1	13	0	
1	11:15	Customer	2	13	0	
1	11:20	Customer	0	16	0	
1	11:23	Customer	0	17	0	
1	11:24	Customer	0	18	0	
1	11:26	Customer	0	19	0	
1	11:30	Customer	0	20	0	
1	11:30	Customer	1	20	0	
1	11:30	Customer	2	20	0	

Output the summary of the simulation: simulation_summary.csv or simulation_summary_v1.csv

A	B	C	D	E	F	G	H
Day	ni	Number_f	Carrier	Carrier_re	Hunger	Hunger_removed	
1	100	100	10	[1, 3, 6, 8,	0	[]	
2	66	66	7	[13, 16, 22	0	[]	
3	35	35	3	[5, 24, 25,	0	[]	
4	23	23	2	[9, 20, 43]	0	[]	
5	20	20	1	[4, 97]	0	[]	
6	18	18	0	[80]	0	[]	
7	17	17	0	[]	0	[]	
8	17	17	0	[]	0	[]	
9	17	17	0	[]	0	[]	
10	17	17	0	[]	0	[]	
11	17	17	0	[]	0	[]	
12	17	17	0	[]	0	[]	
13	17	17	0	[]	0	[]	
14	17	17	0	[]	0	[]	
15	17	17	0	[]	0	[]	
16	17	17	0	[]	0	[]	
17	17	17	0	[]	0	[]	
18	17	17	0	[]	0	[]	
19	17	17	0	[]	0	[]	
20	17	17	0	[]	0	[]	
21	17	17	0	[]	0	[]	

Task 0:

I will choose task 1, task 5, task 3.

Task 1:

You can modify the period in this line in assignment4_basic_task1

```
# main program

# initialization

# setting some default numbers
# finish all default numbers
seed_number = 0
D = 100
N = 100
T_size = 12
T_detail = [2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 4]
p = 0.8
incubation_period = 7
k = 2
customer_list = []
server_list = []
Customer_file = "customer_seed0.csv"
Result_file = "simulation_result.csv"
Summary_file = "simulation_summary.csv"

open("config_seed0.csv", 'a')
```

Here is the old process for testing flu

```
    # step 3: process the tests for flu, remove having
    for item in customer_list:
        if item.carrier == 1:
            if not item.arrival_time == -1:
                #print(item.ID)
                index = int(np.random.uniform(100))
                if index < p*100:
                    carrier_remove.append(item.ID)
                    tmp_carrier.append(item.ID)
                    print(item.ID)
                num_carrier = num_carrier + 1
```

New program changed for task 1

```
    # step 3: process the tests for flu, remove having p
    for item in customer_list:
        if item.carrier == 1:
            if not item.arrival_time == -1:
                #print(item.ID)
                if item.period  >= incubation_period:
                    carrier_remove.append(item.ID)
                    tmp_carrier.append(item.ID)
                    print(item.ID)
                else:
                    item.period = item.period + 1
                num_carrier = num_carrier + 1
```

Experiment result:

If we use the original data, the customer will get infected too fast, then we cant observe the relation clearly, so we limit the number of the customers in each deck, to reduce the rate of infection.

```
# finish all default numbers
seed_number = 0
D = 100
N = 100
T_size = 36
T_detail = [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
p = 0.8
incubation_period = 0
k = 2
```

Only two customers can be in the one table

Also we add some more tables.

incubation_period = 0

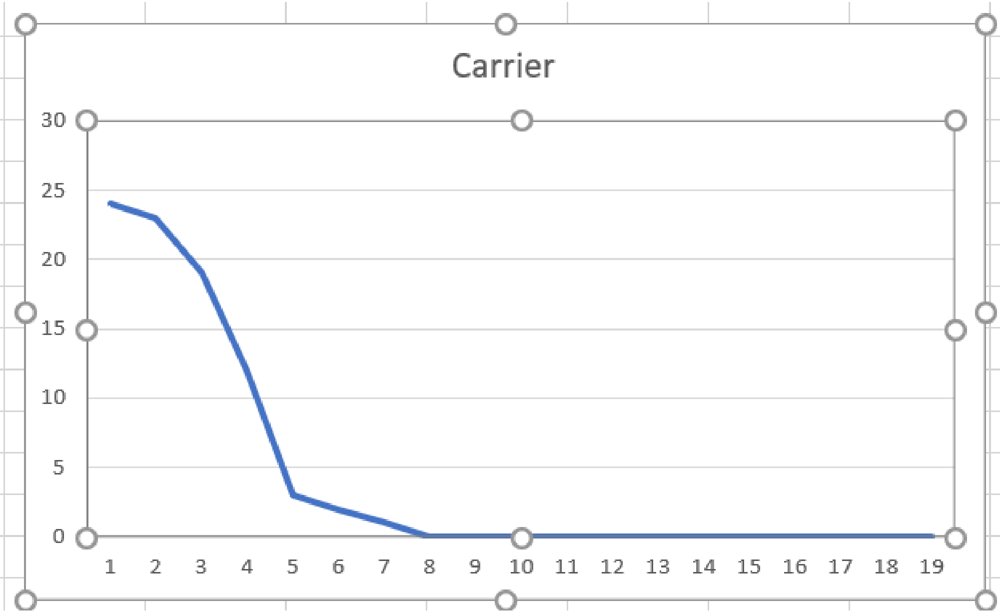
A	B	C	D	E	F	G	H
Day	ni	Number_f	Carrier	Carrier_re	Hunger	Hunger_removed	
1	100	100	0	[1, 2, 3, 11	0	[]	
2	76	76	0	[]	0	[]	
3	76	76	0	[]	0	[]	
4	76	76	0	[]	0	[]	

This is a control experiment, it shows day 1 will infect 24 people.

We can see the infect speed of this set up is slow, so we can observe the relation easily.

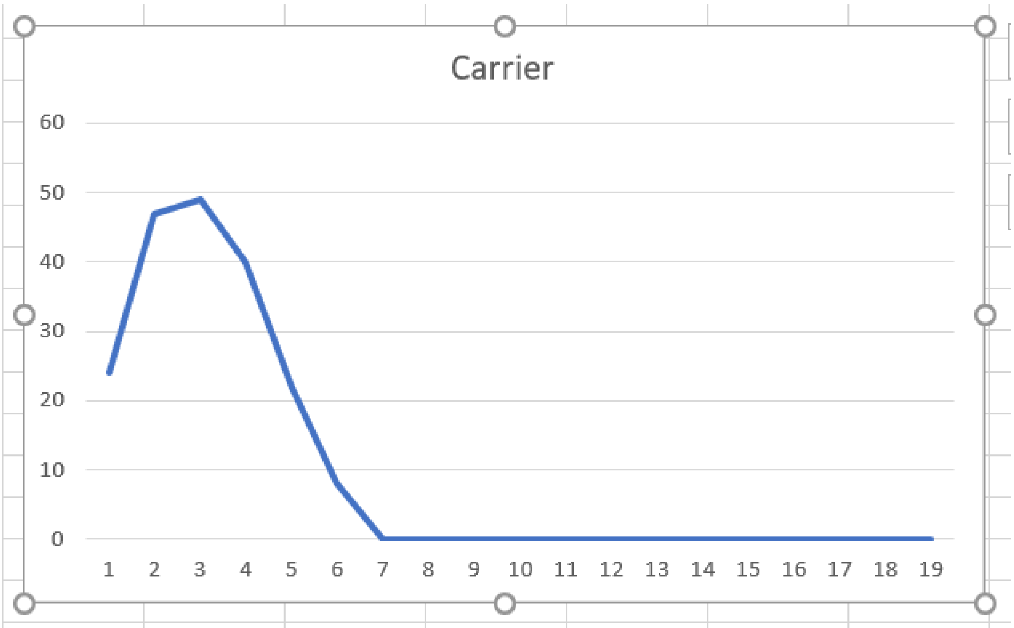
incubation_period = 1

A	B	C	D	E	F	G	H
Day	ni	Number_f	Carrier	Carrier_re	Hunger	Hunger_removed	
1	100	100	24	[]	0	[]	
2	100	100	23	[1, 2, 3, 11	0	[]	
3	76	76	19	[8, 13, 15,	0	[]	
4	53	53	12	[5, 6, 10, 1	0	[]	
5	34	34	3	[9, 12, 16,	0	[]	
6	22	22	2	[47, 52, 92	0	[]	
7	19	19	1	[26, 73]	0	[]	
8	17	17	0	[95]	0	[]	
9	16	16	0	[]	0	[]	
10	16	16	0	[]	0	[]	
11	16	16	0	[]	0	[]	



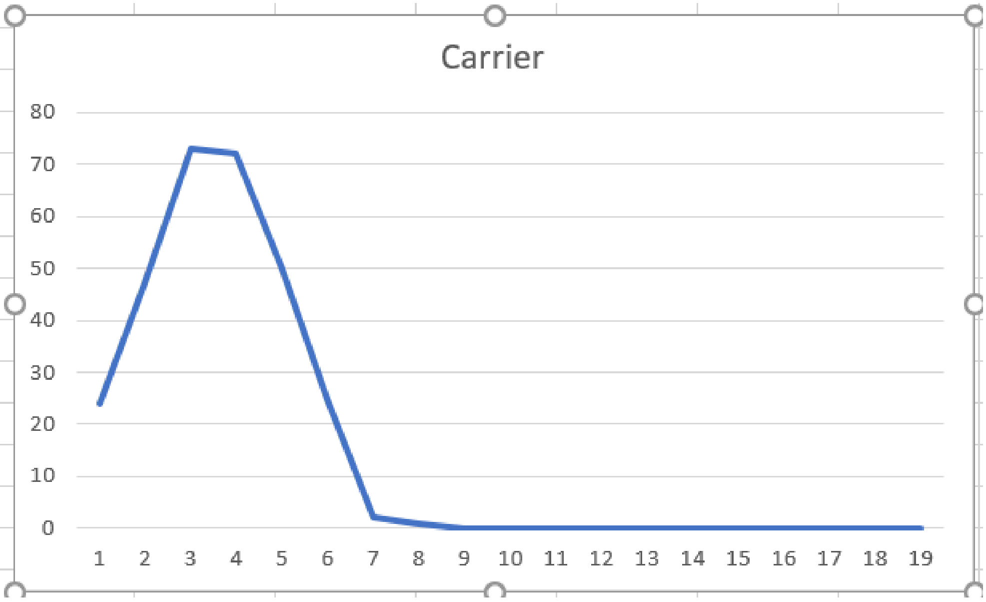
incubation_period = 2

A	B	C	D	E	F	G	H
Day	ni	Number_f	Carrier	Carrier_re	Hunger	Hunger_removed	
1	100	100	24	[]	0	[]	
2	100	100	47	[]	0	[]	
3	100	100	49	[1, 2, 3, 11	0	[]	
4	76	76	40	[8, 13, 15,	0	[]	
5	53	53	22	[4, 5, 10, 1	0	[]	
6	27	27	8	[12, 18, 22	0	[]	
7	13	13	0	[6, 9, 26, 7	0	[]	
8	5	5	0	[]	0	[]	
9	5	5	0	[]	0	[]	
10	5	5	0	[]	0	[]	



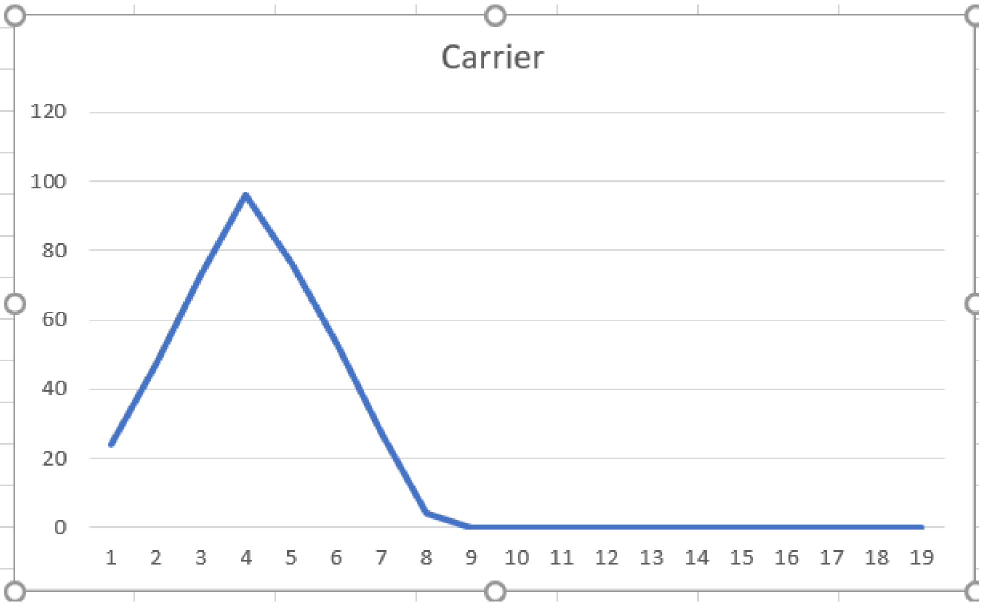
incubation_period = 3

A	B	C	D	E	F	G	H
Day	ni	Number_f	Carrier	Carrier_re	Hunger	Hunger_removed	
1	100	100	24	[]	0	[]	
2	100	100	47	[]	0	[]	
3	100	100	73	[]	0	[]	
4	100	100	72	[1, 2, 3, 11	0	[]	
5	76	76	50	[8, 13, 15,	0	[]	
6	53	53	25	[4, 5, 10, 1	0	[]	
7	27	27	2	[6, 9, 12, 1	0	[]	
8	4	4	1	[55]	0	[]	
9	3	3	0	[7]	0	[]	
10	2	2	0	[]	0	[]	
11	2	2	0	[]	0	[]	
12	2	2	0	[]	0	[]	



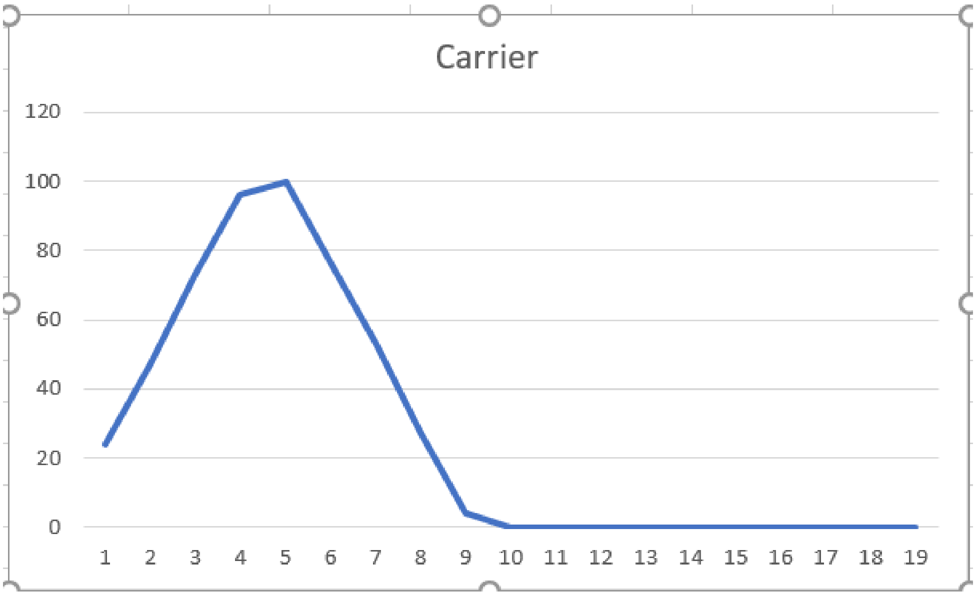
incubation_period = 4

A	B	C	D	E	F	G	H
Day	ni	Number_f	Carrier	Carrier_re	Hunger	Hunger_removed	
1	100	100	24	[]	0	[]	
2	100	100	47	[]	0	[]	
3	100	100	73	[]	0	[]	
4	100	100	96	[]	0	[]	
5	100	100	76	[1, 2, 3, 11	0	[]	
6	76	76	53	[8, 13, 15,	0	[]	
7	53	53	27	[4, 5, 10, 1	0	[]	
8	27	27	4	[6, 9, 12, 1	0	[]	
9	4	4	0	[7, 26, 55,	0	[]	
10	0	0	0	[]	0	[]	
11	0	0	0	[]	0	[]	
12	0	0	0	[]	0	[]	



incubation_period = 5

Day	ni	Number_f	Carrier	Carrier_re	Hunger	Hunger_removed
1	100	100	24	[]	0	[]
2	100	100	47	[]	0	[]
3	100	100	73	[]	0	[]
4	100	100	96	[]	0	[]
5	100	100	100	[]	0	[]
6	100	100	76	[1, 2, 3, 11	0	[]
7	76	76	53	[8, 13, 15,	0	[]
8	53	53	27	[4, 5, 10, 1	0	[]
9	27	27	4	[6, 9, 12, 1	0	[]
10	4	4	0	[7, 26, 55,	0	[]
11	0	0	0	[]	0	[]
12	0	0	0	[]	0	[]
13	0	0	0	[]	0	[]



The graph are showing the number of carrier in the canteen on each day.

After testing incubation_period 1-5, we can clearly see that with smaller incubation_period, the spread of the flu will be smaller and the final number of infected customer is fewer.

We can see for incubation_period 1-3,the final number of customer got infected (remain 16,5,2 respectively) decreases when incubation_period increases, the peak of the number of carrier in the first 20 days (24,49,73 respectively) increase with the incubation_period.

The slope of the graph shows the rate of the spread of the flu, the higher is the slope, the flu will spreads faster, as we know the more infected customer exists at the same time, the flu will spread much more faster.

So if the incubation_period is smaller, then we can blacklist the infected customers earlier, then we can reduce the peak of the number of carrier and the slope of the graph earlier, before the peak got much higher.

Therefore, we can conclude that the speed of the spread of the flu will increases with the length of the incubation period.

Task 5:

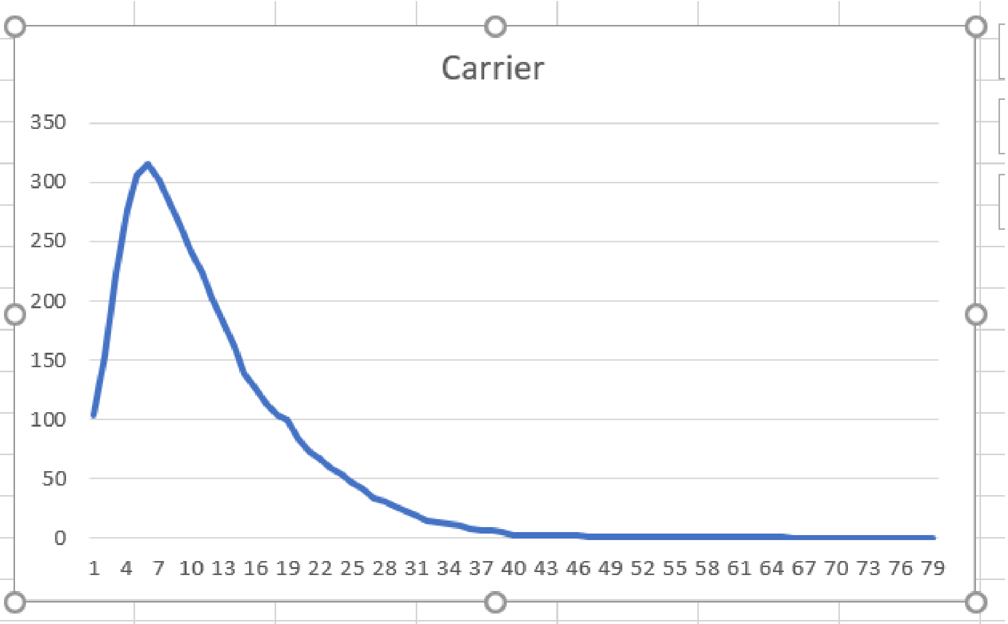
The steady state is the constant or steady state that is established after a certain time in the system or simulation. The transient state means the state between the beginning of the event and the steady state, the value in this state is still varying, and it will become steady and reach steady state after a certain time later.

I really don't understand what the question means "convert to different steady states after 50 days", so I need to have steady state before or after day 50? Or I just need to make a graph with steady state? So I try to make some strange steady states which can fulfil all conditions.

Experiment one:

```
# setting some default numbers
# finish all default numbers
seed_number = 0
D = 100
N = 500
T_size = 18
T_detail = [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
p = 0.1
k = 50
customer_list = []
server_list = []
Customer_file = "customer_seed0.csv"
Result_file = "simulation_result.csv"
Summary_file = "simulation_summary.csv"
```

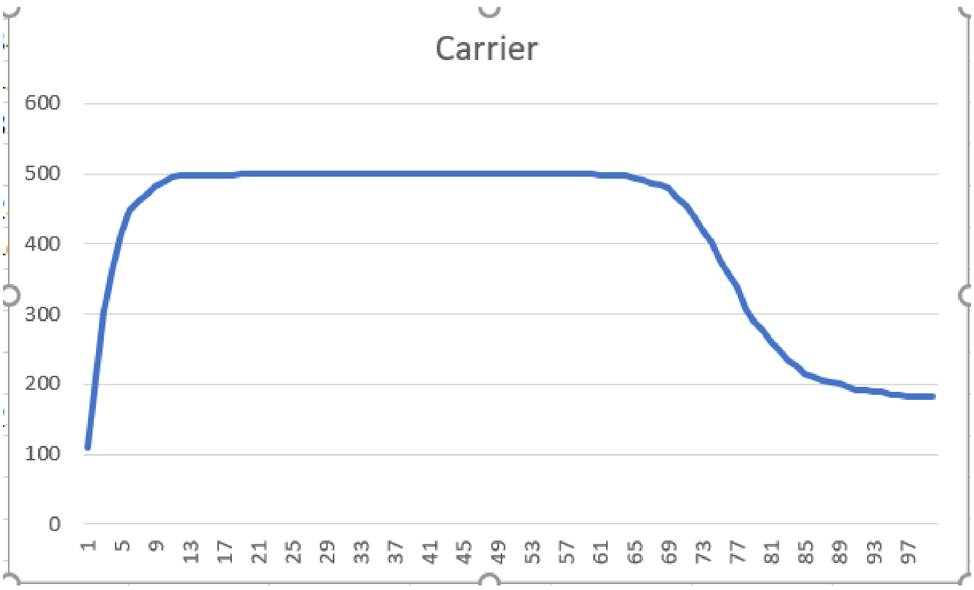
Value of k can be ignore, as it cant affect the result here.



We can see here, we have extreme low p, the sharp transient state is at around day 1 to day 30 as all customer got infected and then got blacklisted slowly after that, then finally there is no customer, it gradually convert to steady state after day 50.

Experiment two:

```
seed_number = 0
D = 100
N = 500
T_size = 18
T_detail = [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
p = 0
k = 50
customer_list = []
server_list = []
Customer_file = "customer_seed0.csv"
Result_file = "simulation_result.csv"
Summary_file = "simulation_summary.csv"
```



In this experiment, we would not blacklist customer, but we will have extreme large number of customers, and they can be hunger for 50 days! So we can make sure we will have two transient state and two steady state in this experiment.

At day 1 to 10, customer starts to infect flu, this is a transient state, then at day 13 and day 65, most of the customers infect flu, then we have a first long steady state. Until around day 69, many hungry customers will be angry and will not come again, then the experiment will experience second huge change, then we have a transient state at day 69 to 80, then after many hungry customers leave, then we will convert to the second steady state after day 90. After that it should be steady forever (as this simulation is too long, I cant show it more after day 100).

In order attain steady state after day 50, I try to use some extreme input, so it can extent the process of transient state, and slow down the process to steady state.

In conclusion, in order to make transient state, we need to add some change to the system, then the system will process the change and finally reach the steady state, so I try to add some change after day 50 or some change can still affect the system after day 50, to make sure the steady states are located after day 50.

Task 3:

I try to make the program will count money, each customer will give 1 coin after dined.

```
Departure time =  145 Customer ID =  35 Table =  10 curr_time =  145
Departure time =  156 Customer ID =  59 Table =  4 curr_time =  156
money =  1202

C:\Users\CUA\Desktop\IERG3050_2021Spring_Assignment4_1155126366>
```

T_size = 18

T_detail = all 1 in []

For using all three 2-person tables

```
Departure time =  157 Customer ID =  20 Table =  2 curr_time =  1
Departure time =  161 Customer ID =  38 Table =  10 curr_time =
Departure time =  176 Customer ID =  28 Table =  13 curr_time =
Departure time =  196 Customer ID =  53 Table =  11 curr_time =
Departure time =  204 Customer ID =  73 Table =  18 curr_time =
money =  4748

C:\Users\CUA\Desktop\IERG3050_2021Spring_Assignment4_1155126366>
```

T_size = 12

T_detail = [4,4,4,4,4,4,4,4,4,4,4,4]

For using all Two 4-person tables

```
Departure time =  126 Customer ID =  90 Table =  2 curr_time =  12
Departure time =  132 Customer ID =  72 Table =  11 curr_time =  1
Departure time =  138 Customer ID =  93 Table =  3 curr_time =  13
Departure time =  140 Customer ID =  24 Table =  4 curr_time =  14
Departure time =  148 Customer ID =  44 Table =  9 curr_time =  14
Departure time =  156 Customer ID =  59 Table =  10 curr_time =  1
money =  1959

C:\Users\CUA\Desktop\IERG3050_2021Spring_Assignment4_1155126366>
```

So I will use all three 2-person tables, we can make more money