I. A brief description of how you conducted the experiment.

A. Firstly, I found a thousand additional domains from the web browser, and aggregated them with the 4,000 domains provided by the tutor at forum, totalling 5,092 in a .txt file.

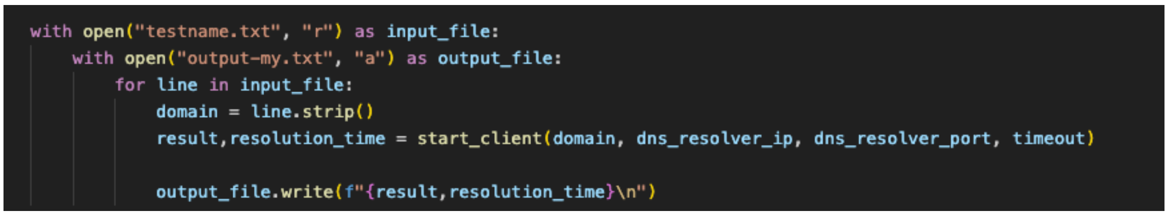
B. I imported the time library in python, set a start time before the start of the establishment time on the client side, and an end time after the single connection is closed, then calculate the intermediate time named solution time. and use it to measure and compare the performance of the resolver.

C. I used. write() method to automatically store the returned solution time after each domain name is sent in a txt record, and wait until all domain names have been run. If the query fails, then an error message is stored instead of solution time.

II. A brief description of how the experiment can be repeated.

To conduct the experiment automatedly, I slightly modified my client and resolver code to use.

A. The client side uses two nested for loops, the outer for loop opens the input file, which is automatically tested for each domain name prepared, and the inner for loop opens the output file, which records the solution time for each query. Before the end of the loop, the experiment is automatically repeated.

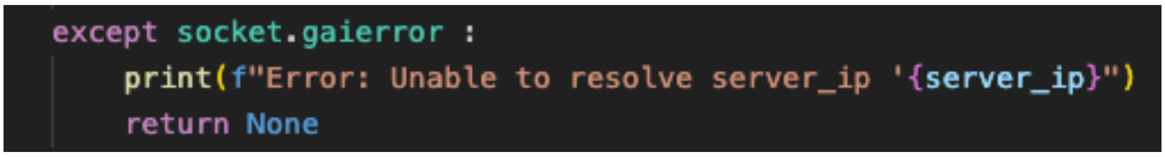


```python
with open("testname.txt", "r") as input_file:
    with open("output-my.txt", "a") as output_file:
        for line in input_file:
            domain = line.strip()
            result,resolution_time = start_client(domain, dns_resolver_ip, dns_resolver_port, timeout)

            output_file.write(f"{result,resolution_time}\n")
```

B. Maintained the previous basic condition that the resolver is in the listening state unless it is manually shut down by users.

C. Added some try-excepts to catch various errors in the send and parse query of the resolver, to make sure that the connection is not broken even if certain errors occur. For example, socket.gaierror is triggered when a given hostname or service name cannot be resolved.



```python
except socket.gaierror :
    print(f"Error: Unable to resolve server_ip '{server_ip}")
    return None
```

III. The results presented in tabular form.

A. The rate of success query, which shows the 8.8.8.8 is the best, second is the 1.1.1.1, it's a little sat my resolver is the last but it makes sense.

total nubmer of domain	5902		
\	success query	error	rate of success query
my-resolver	5522	380	93.56%
8.8.8.8	5861	41	99.31%
1.1.1.1	5835	67	98.86%

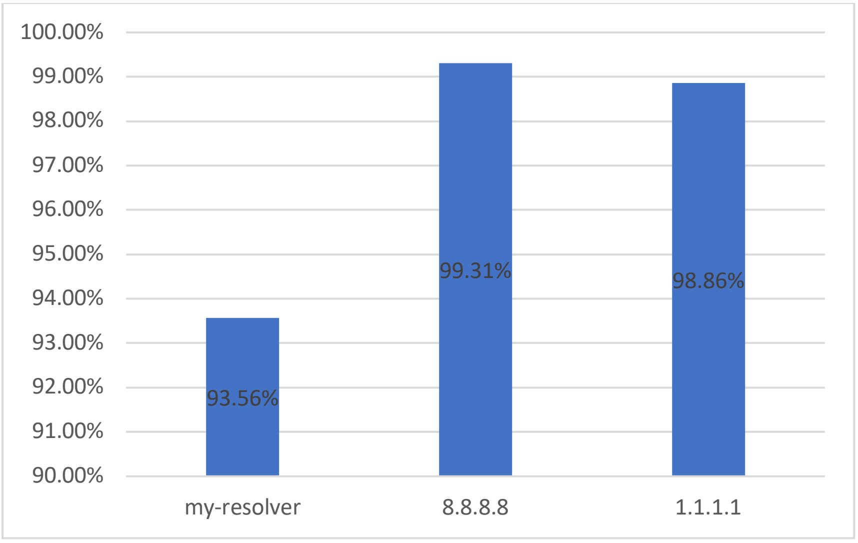
B. This is the total time of all queries (5092). We can see the 8.8.8.8 is the best and much even better than another two.

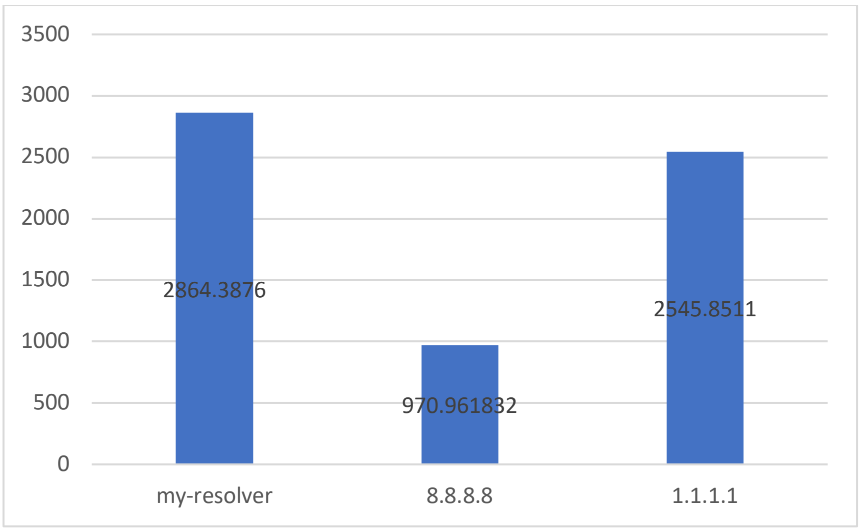
\	total time of all queries(seconds)
my-resolver	2864.39
8.8.8.8	970.96
1.1.1.1	2545.85

C. This is the average time and the value of variance. It shows that the 8.8.8.8 is the most stable resolver.

	average time	value of variance
my-resolver	0.608	0.231
8.8.8.8	0.190	0.106
1.1.1.1	0.507	0.236

IV. The results presented in graphical form.




3500
3000
2500
2000
1500
1000
500
0
2864.3876
970.961832
2545.8511
my-resolver
8.8.8.8
1.1.1.1