1. What have I done in this assignment:

   I have finished all the requirement as described in the Project requirement.

   - In Phase1A, all four servers will up and run. And the client will get the function and input from command line and send them to the AWS server over TCP connection.
   - In Phase1B, the AWS server will send the function and input to the 3 back-servers using UDP connection. Each back-server will perform their respective database operation and then return their result to the AWS server using UDP.
   - In Phase2, the AWS server should combine the results collected from the 3 back-servers and send the results to both client and monitor.

2. what each file does:

   - serverA.cc: After it get data from AWS using UDP, it will perform its respective database operation and then send back the result to AWS.
   - serverB.cc: After it get data from AWS using UDP, it will perform its respective database operation and then send back the result to AWS.
   - serverC.cc: After it get data from AWS using UDP, it will perform its respective database operation and then send back the result to AWS.
   - aws.cc: it will get the function and input from client and then send them to 3 back-servers. After getting result from 3 back-servers, it will combine the results and send the final result to client and monitor.
   - client.cc: it will get the function and input from command line and send to AWS server using TCP connection. After processing, it will get the result and show on the screem.
   - monitor.cc: After processing, it will get the result from the AWS server and show on the screem.

3. how to run the programs:

   i. Frist you should open six different terminals. And using one of them to type `make all` to compile all the file.
   ii. typing `./bin/serverA.out` using one terminal to run serverA.
   iii. typing `./bin/serverB.out` using one terminal to run serverB.
   iv. typing `./bin/serverC.out` using one terminal to run serverC.
   v. typing `./bin/aws.out` using one terminal to run AWS server.
   vi. typing `./bin/monitor.out` using one terminal to run monitor.
   vii. typing `./bin/client.out <function> <input>` using one terminal to run client. the must be one of 'search' and 'prefix' and the must be a word consisting of 27 characters or less.

4. The format of all the messages exchanged:

   ./bin/client.out prefix Sea:

```
The client is up and running.
The client sent <Sea> and <prefix> to AWS.
Found <36> match(es) for <Sea>:
<Seam>
<Sea fern>
<Sea grape>
<Sea chickweed>
<Sea orange>
<Sea bank>
<Sea mat>
<Sea letter>
<Sea onion>
<Seam>
```

```
<Sea cow>
<Sea perch>
<Sea dragon>
<Sea bass>
<Seabound>
<Sea poppy>
<Sea lily>
<Sea monster>
<Sea butterfly>
<Seamed>
<Seaboat>
<Sea cock>
<Sea bean>
<Sea king>
<Sea pie>
<Sea perch>
<Sea gull>
<Sea peach>
<Sea holly>
<Sea poker>
<Sea louse>
<Sea elephant>
<Seamanship>
<Sea devil>
<Sea mouse>
<Sea owl>
```

./bin/aws.out:

```
The AWS is up and running.
The AWS received input=<Sea> and function=<prefix> from the client using TCP over port <25146>.
The AWS sent <Sea> and <prefix> to Backend-Server A.
The AWS received <12> match(es) from Backend-Server <A> using UDP over port <21146>.
The AWS sent <Sea> and <prefix> to Backend-Server B.
The AWS received <14> match(es) from Backend-Server <B> using UDP over port <22146>.
The AWS sent <Sea> and <prefix> to Backend-Server C.
The AWS received <10> match(es) from Backend-Server <C> using UDP over port <23146>.
The AWS sent <36> matches to client.
The AWS sent <36> matches to the monitor via TCP port 26146.
```

./bin/serverA.out:

```
The ServerA is up and running using UDP on port<21146>.
The ServerA received input <Sea> and operation <prefix>.
The ServerA has found <12> matches.
The ServerA finished sending the output to AWS.
```

./bin/serverB.out:

```
The ServerB is up and running using UDP on port <22146>.
The ServerB received input <Sea> and operation <prefix>.
The ServerB has found <14> matches.
The ServerB finished sending the output to AWS.
```

./bin/serverC.out:

```
The ServerC is up and running using UDP on port <23146>.
The ServerC received input <Sea> and operation <prefix>.
The ServerC has found <10> matches.
The ServerC finished sending the output to AWS.
```

./bin/monitor.out:

```
The monitor is up and running.
Found <36> match(es):
<Seam>
<Sea fern>
<Sea grape>
<Sea chickweed>
<Sea orange>
<Sea bank>
<Sea mat>
<Sea letter>
<Sea onion>
<Seam>
<Sea cow>
<Sea perch>
<Sea dragon>
<Sea bass>
<Seabound>
<Sea poppy>
<Sea lily>
<Sea monster>
<Sea butterfly>
<Seamed>
<Seaboat>
<Sea cock>
<Sea bean>
<Sea king>
<Sea pie>
<Sea perch>
<Sea gull>
<Sea peach>
<Sea holly>
<Sea poker>
<Sea louse>
<Sea elephant>
<Seamanship>
<Sea devil>
<Sea mouse>
<Sea owl>
```

5. the idiosyncrasy:

   Under my test case, I have net found any fail. However, to run the project, you must obey the order described in "how to run" part.

6. Reused Code:

   There is no any reused code from anywhere in the program.