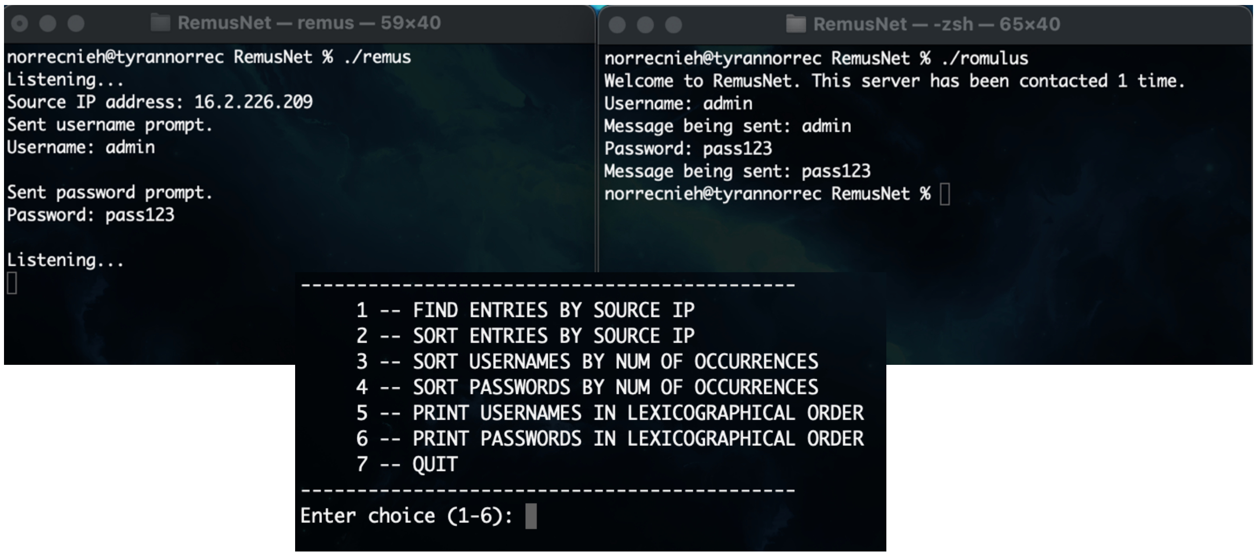
Norrec Nieh

CS 5008 Final Capstone Increment Specifications

Increment 1: **Building a Server**

- **Get server (remus.c) working.**
- Server should be able to get the client-side IP address from each connection.
- Server should also be able to get a username and password from the client.
- Implement menu structure for log parser program.



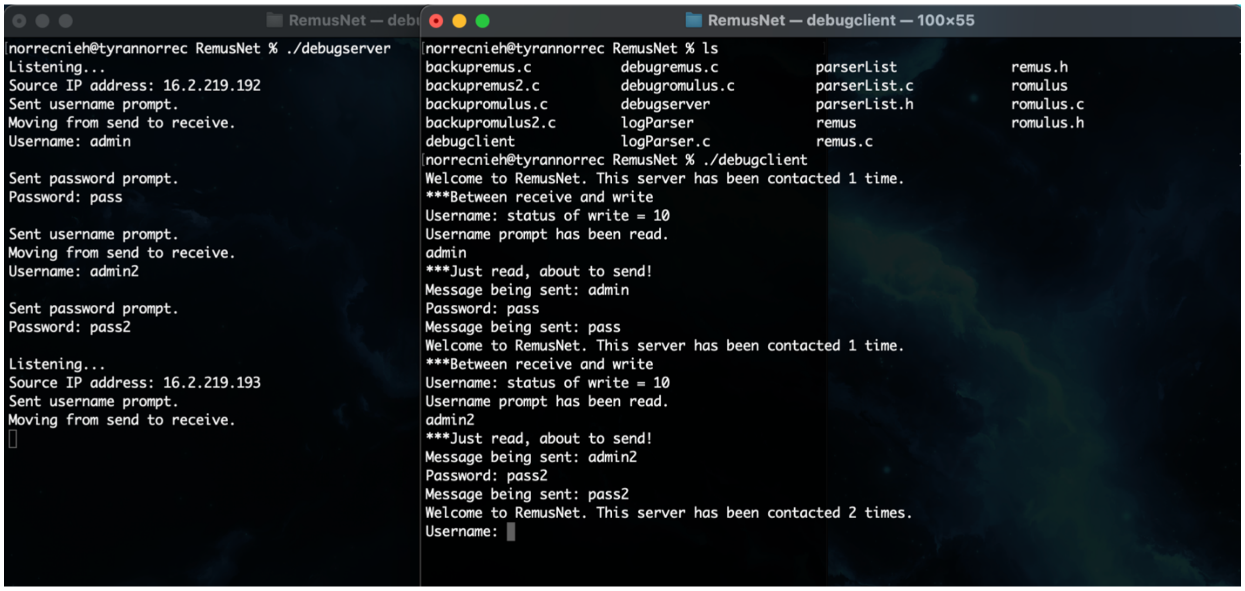
```
norrecnieh@tyrannorrec RemusNet % ./remus
Listening...
Source IP address: 16.2.226.209
Sent username prompt.
Username: admin

Sent password prompt.
Password: pass123

Listening...
```

```
norrecnieh@tyrannorrec RemusNet % ./romulus
Welcome to RemusNet. This server has been contacted 1 time.
Username: admin
Message being sent: admin
Password: pass123
Message being sent: pass123
norrecnieh@tyrannorrec RemusNet %
```

```
--------------------------------------------
     1 -- FIND ENTRIES BY SOURCE IP
     2 -- SORT ENTRIES BY SOURCE IP
     3 -- SORT USERNAMES BY NUM OF OCCURRENCES
     4 -- SORT PASSWORDS BY NUM OF OCCURRENCES
     5 -- PRINT USERNAMES IN LEXICOGRAPHICAL ORDER
     6 -- PRINT PASSWORDS IN LEXICOGRAPHICAL ORDER
     7 -- QUIT
--------------------------------------------
Enter choice (1-6):
```

Increment 2: **Client-Server Communication**

- **Get client (romulus.c) working with the server (remus.c).**
- Implement username/password logger function and be able to call on a loop.
- Loop the server-client communication so that credential exchange can be repeated.
- Debug server-client interaction and resolve blocking issue.

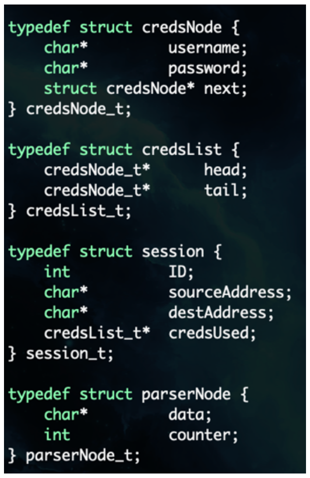


```
norrecnieh@tyrannorrec RemusNet % ./debugserver
Listening...
Source IP address: 16.2.219.192
Sent username prompt.
Moving from send to receive.
Username: admin

Sent password prompt.
Password: pass

Sent username prompt.
Moving from send to receive.
Username: admin2

Sent password prompt.
Password: pass2

Listening...
Source IP address: 16.2.219.193
Sent username prompt.
Moving from send to receive.
```

```
norrecnieh@tyrannorrec RemusNet % ls
backupremus.c           debugremus.c            parserList              remus.h
backupremus2.c          debugromulus.c          parserList.c            romulus
backupromulus.c         debugserver             parserList.h            romulus.c
backupromulus2.c        logParser               remus                   romulus.h
debugclient             logParser.c             remus.c
norrecnieh@tyrannorrec RemusNet % ./debugclient
Welcome to RemusNet. This server has been contacted 1 time.
***Between receive and write
Username: status of write = 10
Username prompt has been read.
admin
***Just read, about to send!
Message being sent: admin
Password: pass
Message being sent: pass
Welcome to RemusNet. This server has been contacted 1 time.
***Between receive and write
Username: status of write = 10
Username prompt has been read.
admin2
***Just read, about to send!
Message being sent: admin2
Password: pass2
Message being sent: pass2
Welcome to RemusNet. This server has been contacted 2 times.
Username:
```

Increment 3: **Data Structures and Essential Methods**

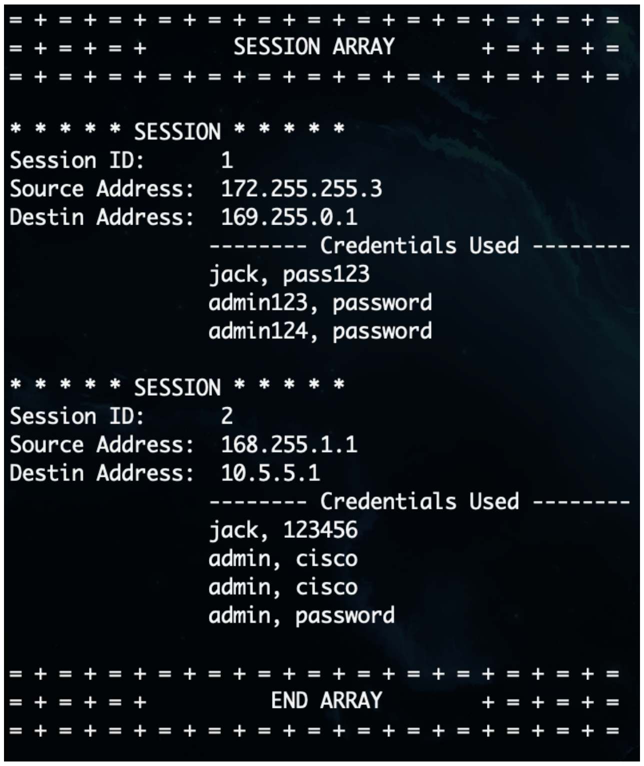
- **parser.h should have essential structs defined.**
- **parser.c should have essential list methods implemented**, including sets of methods for creating, printing, and deleting nested structs, as well as variations for node insertion.
- Define and implement a separate struct (parserNode_t) and associated methods to extract usernames and passwords from the session array for further manipulation. These relate to menu options 3 to 6.
- At this point, the parser is being tested in main() of parserList.c



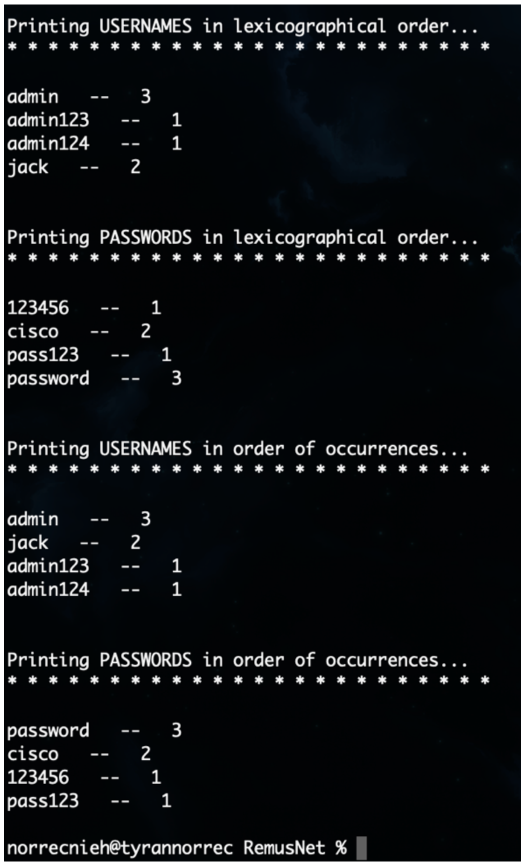
```
typedef struct credsNode {
    char*           username;
    char*           password;
    struct credsNode* next;
} credsNode_t;

typedef struct credsList {
    credsNode_t*     head;
    credsNode_t*     tail;
} credsList_t;

typedef struct session {
    int             ID;
    char*           sourceAddress;
    char*           destAddress;
    credsList_t*    credsUsed;
} session_t;

typedef struct parserNode {
    char*           data;
    int             counter;
} parserNode_t;
```

Increment 4: **Parsing Credentials**

- **Implement mergeSort** for parserNode_t structs; sorts are done lexicographically on contents of data attribute of parserNode_t struct, containing usernames or passwords depending on the array.
- **Implement binary search** for parserNode_t arrays.
- **Implement functions for menu options 3 to 6**.
- Use the mergesort and binary search functions, as well as functions delineated in increment 3, to implement all methods relating to username and password arrays:
    - o printUserArrayLex();   // print all usernames by lexicographical order
    - o printPassArrayLex();   // print all passwords by lexicographical order
    - o printUserArrayOcc();   // mergesort all usernames by non-increasing num of occurrences
    - o printPassArrayOcc();   // mergesort all passwords by non-increasing num of occurrences



```
= + = + = + = + = + = + = + = + = + = + = + = + =
= + = + = +       SESSION ARRAY       + = + = + =
= + = + = + = + = + = + = + = + = + = + = + = + =

* * * * * SESSION * * * * *
Session ID:      1
Source Address:  172.255.255.3
Destin Address:  169.255.0.1
                 -------- Credentials Used --------
                 jack, pass123
                 admin123, password
                 admin124, password

* * * * * SESSION * * * * *
Session ID:      2
Source Address:  168.255.1.1
Destin Address:  10.5.5.1
                 -------- Credentials Used --------
                 jack, 123456
                 admin, cisco
                 admin, cisco
                 admin, password

= + = + = + = + = + = + = + = + = + = + = + = + =
= + = + = +          END ARRAY        + = + = + =
= + = + = + = + = + = + = + = + = + = + = + = + =
```



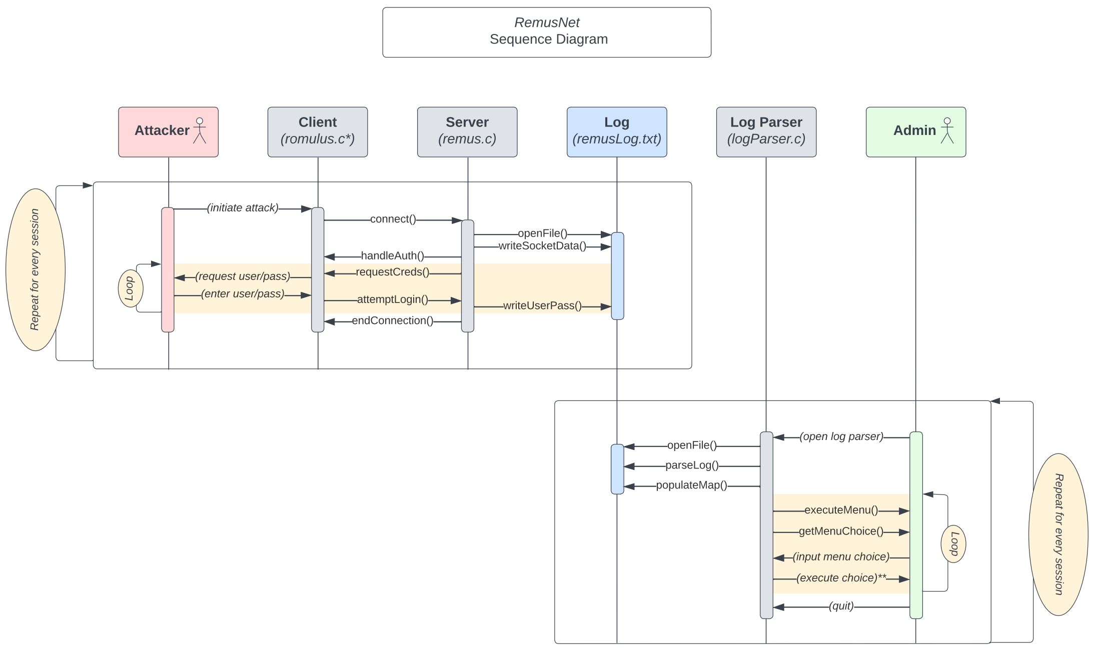
```
Printing USERNAMES in lexicographical order...
* * * * * * * * * * * * * * * * * * * * * * * *

admin   --   3
admin123   --   1
admin124   --   1
jack   --   2


Printing PASSWORDS in lexicographical order...
* * * * * * * * * * * * * * * * * * * * * * * *

123456   --   1
cisco   --   2
pass123   --   1
password   --   3


Printing USERNAMES in order of occurrences...
* * * * * * * * * * * * * * * * * * * * * * * *

admin   --   3
jack   --   2
admin123   --   1
admin124   --   1


Printing PASSWORDS in order of occurrences...
* * * * * * * * * * * * * * * * * * * * * * * *

password   --   3
cisco   --   2
123456   --   1
pass123   --   1

norrecnieh@tyrannorrec RemusNet %
```

Increment 5: **Application and Menu Implementation**

- **Implement populate() in parser.c; test with mock log file**
- **Implement functions for menu options 1 and 2**
    - Find Entries by Source IP
    - Sort Entries by Source IP
        - Write functions to sort IP addresses, including parsing the string holding the IP addresses and padding with 0's in order to compare lexicographically
- **All functions should be integrated via menu options in parserMenu.c**
- Implement extra minor functions if time permits:
    - Built-in timer during mergesort for complexity analysis
    - Manually Add Session
    - Find and Delete Session by ID
    - Find Entries by Session ID
    - Sort Entries by Session ID
    - Binary Search for num of occurrences of particular username/password
- At this point, **parserMenu.c** should be fully functional with all menu options implemented.

Increment 6: **Finishing the Server and Client Programs**

- Further debug server and client.
- **Output process to log file** should be designed and formatted properly.
- Refactor existing code in the server and client into functions, if possible.
- Run and collect data for parsing; use to test parser.



* For the purpose of simulation, our own client will be used by the "attacker".
** Menu choices expressed here as a single step for simplicity's sake.
See the following diagrams for details.