1)
I learned about coding a TCP socket where we made a server and clients that communicated. I first learned about TCP, where to connect as a client, you have to send a packet, the server sends an acknowledgment packet, and the client sends the acknowledgment process. Learn how the server is created by making the socket, binding it with the address information, and then listening. When a client connects it, it accepts where it reviews the client's address information. Then, it reads for any packet; if it reads anything, it writes to the client back with it received client's address information from the accept function.

Then I learned about the <sys/socket.h> library where we use function socket() to create a socket with TCP confirmation (parameters for TCP in the socket). Then I learned about structures again in c and learned about sockarrd_in, where it's used to do binding in sever's code. While in the clients, the binding uses sockaddr struct instead of sockarrd_in, and I learned about how structs can be cast too.

Then, I learned about uptime and how to capture it where. I learned a lot of string manipulation functions, strcat, and other functions that I didn't know; instead, I used sprint since it's the easiest.I also needed help with the compiler not recognizing where the sys/socket was.h as I learned to include the path.

2)
One approach is use popen() function when running this

```
FILE *fp;
fp = popen("uptime","r" );
if(fp ==NULL){
    perror("uptime commad failed to read");
    exit(1);
}
char writebuffer_tem[buffersize];
fgets(writebuffer_tem,buffersize, fp);
writebuffer_tem[strlen(writebuffer_tem)-1]='\0';
pclose(fp);
```

Returns a FILE type with uptime commands output. Then reading the FILE value you can use fgets() function to get the content of the file value up to a given size.

The second approach is to use the system function to run the command uptime, so it saves to a file by puting “>” like this “uptime > {file location}”:

```
FILE *fp;
char command[1024];
char temp_file[] = "/tmp/cpre489";

mkstemp(temp_file);
snprintf(command, sizeof(command), "uptime > %s", temp_file);
system(command);

fp = fopen(temp_file, "r");
if (fp) {
    char line[1024];
    while (fgets(line, sizeof(line), fp)) {
        printf("%s", line);
    }
    fclose(fp);
}
remove(temp_file);
```

Mkstamp creates the file in temp_file and the system call that the command from the sprintf’s result then it fopens the tem file reads to the desired string and closes the files .