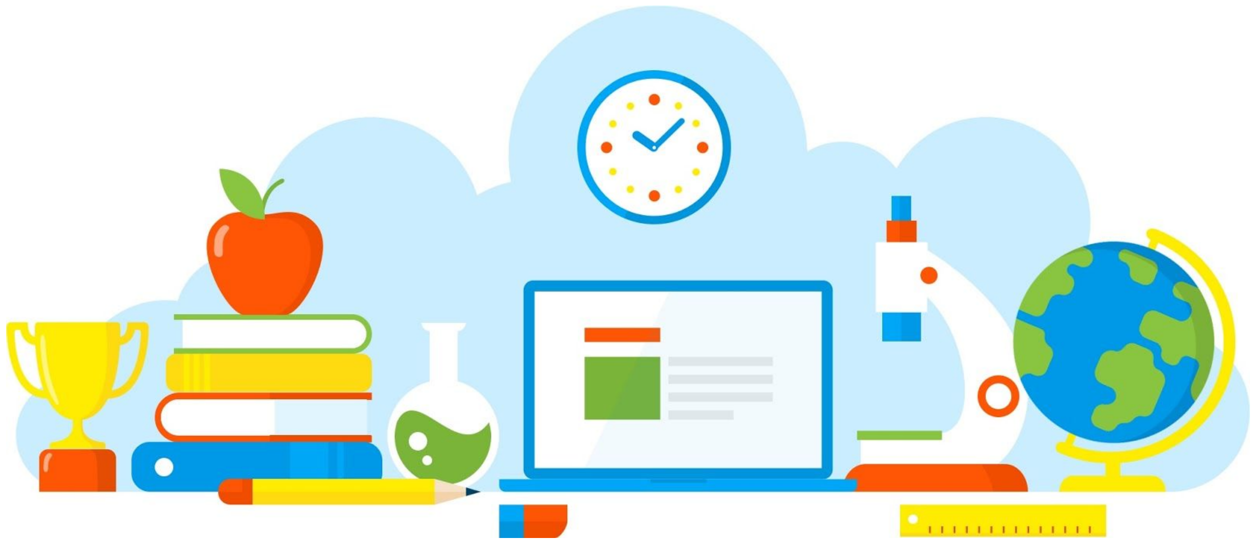


Assignment Report

This report contains work from Assignment 1 of course CSN 361.

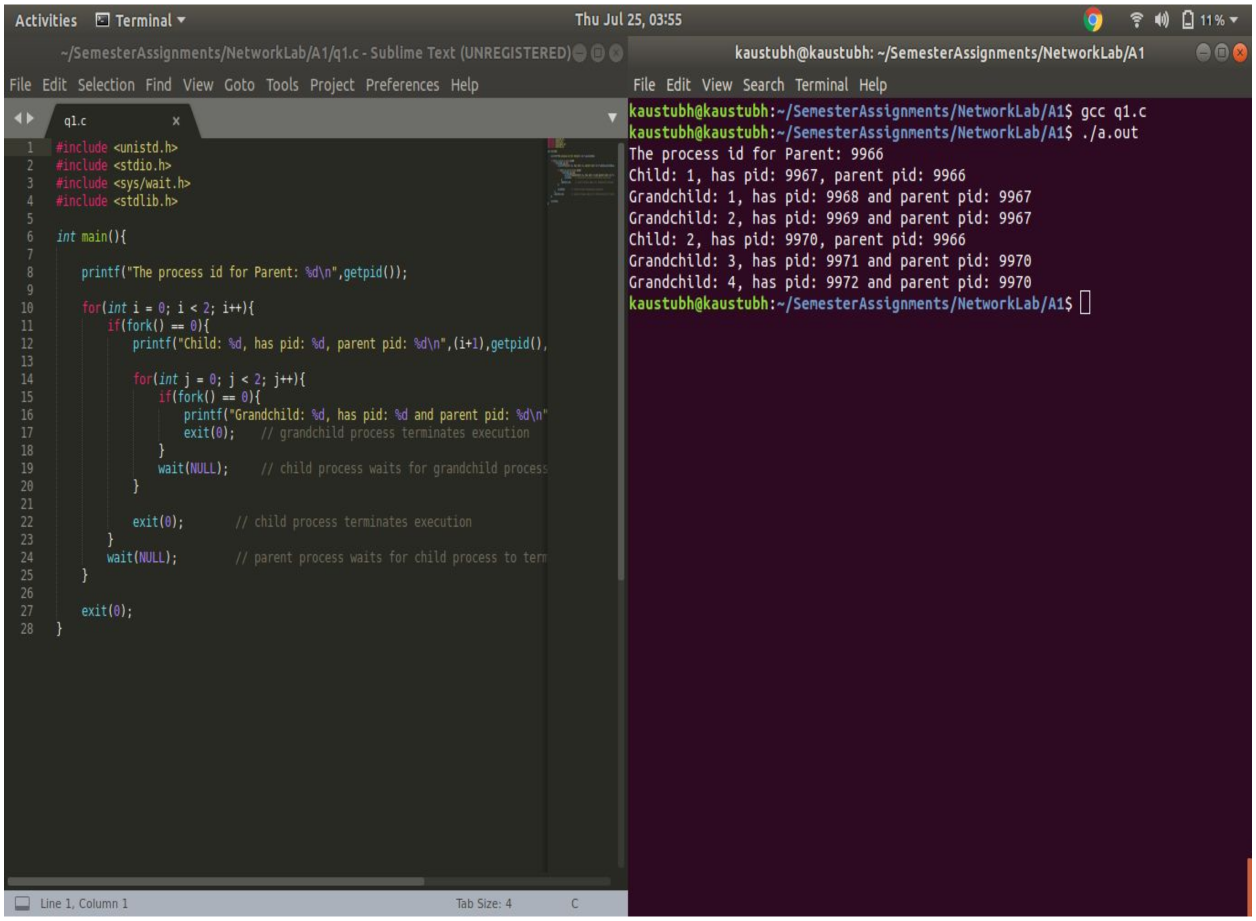
Name- Kaustubh Trivedi
Enrollment Number - 17114044
Class - CSE B.Tech. 3rd Year
Submission Files - [Repository_Link]

QUESTION 1: Write a C program in UNIX system that creates two children and four grandchildren (two for each child). The program should print the process id of the two children, the four grandchildren and the parent in this order.

SOLUTION CODE:

```c
#include <unistd.h>
#include <stdio.h>
#include <sys/wait.h>
#include <stdlib.h>

int main(){

      printf("The process id for Parent: %d\n",getpid());

      for(int i = 0; i < 2; i++){
            if(fork() == 0){
                  printf("Child: %d, has pid: %d,
                        parent pid: %d\n",(i+1),getpid(),getppid());

                  for(int j = 0; j < 2; j++){
                        if(fork() == 0){
                              printf("Grandchild: %d, has pid: %d and
                               paren tpid: %d\n",(i*2 + 1 + j),
                               getpid(),getppid());
                              exit(0);
                        }
                        wait(NULL);

                  exit(0);

            wait(NULL);
      }

      exit(0);
}
```

SCREENSHOT:


Activities
Terminal
Thu Jul 25, 03:55
11 %
~/SemesterAssignments/NetworkLab/A1/q1.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
q1.c
kaustubh@kaustubh: ~/SemesterAssignments/NetworkLab/A1
File Edit View Search Terminal Help
Line 1, Column 1
Tab Size: 4
C

```c
#include <unistd.h>
#include <stdio.h>
#include <sys/wait.h>
#include <stdlib.h>

int main(){

    printf("The process id for Parent: %d\n",getpid());

    for(int i = 0; i < 2; i++){
        if(fork() == 0){
            printf("Child: %d, has pid: %d, parent pid: %d\n",(i+1),getpid(),

            for(int j = 0; j < 2; j++){
                if(fork() == 0){
                    printf("Grandchild: %d, has pid: %d and parent pid: %d\n"
                    exit(0);    // grandchild process terminates execution
                }
                wait(NULL);     // child process waits for grandchild process
            }

            exit(0);        // child process terminates execution
        }
        wait(NULL);         // parent process waits for child process to term
    }

    exit(0);
}
```

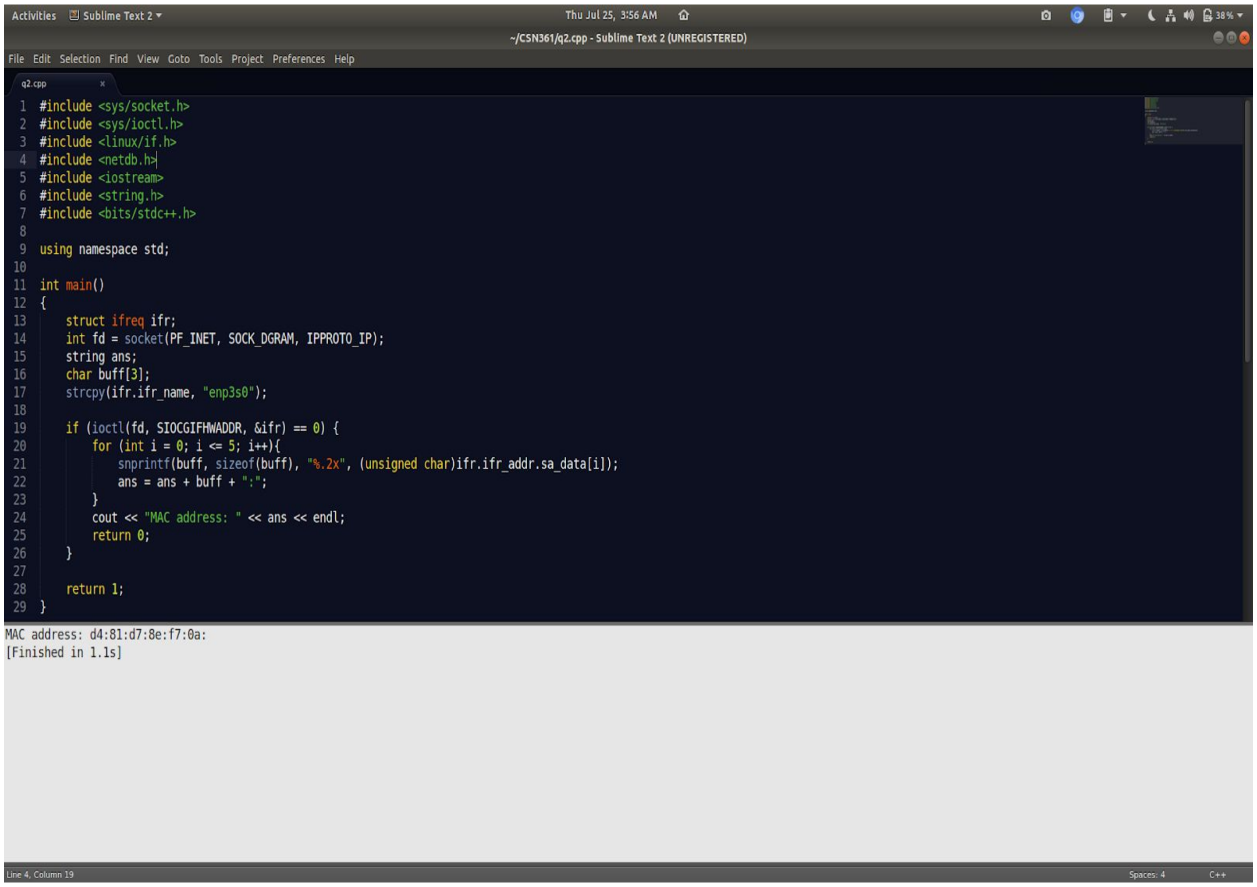
```
kaustubh@kaustubh:~/SemesterAssignments/NetworkLab/A1$ gcc q1.c
kaustubh@kaustubh:~/SemesterAssignments/NetworkLab/A1$ ./a.out
The process id for Parent: 9966
Child: 1, has pid: 9967, parent pid: 9966
Grandchild: 1, has pid: 9968 and parent pid: 9967
Grandchild: 2, has pid: 9969 and parent pid: 9967
Child: 2, has pid: 9970, parent pid: 9966
Grandchild: 3, has pid: 9971 and parent pid: 9970
Grandchild: 4, has pid: 9972 and parent pid: 9970
kaustubh@kaustubh:~/SemesterAssignments/NetworkLab/A1$ 
```

QUESTION 2: Write a C++ program to print the MAC address of your computer.

SOLUTION CODE:

```
#include <sys/ioctl.h>
#include <linux/if.h>
#include <netdb.h>
#include <iostream>
#include <string.h>
#include <bits/stdc++.h>

using namespace std;

int main()
{
    struct ifreq ifr;
    int fd = socket(PF_INET, SOCK_DGRAM, IPPROTO_IP);
    string result;
    char buffer[3];
    strcpy(ifr.ifr_name, "enp3s0");

    if (ioctl(fd, SIOCGIFHWADDR, &ifr) == 0) {
        for (int i = 0; i <= 5; i++){
            snprintf(buffer, sizeof(buffer), "%.2x",
                (unsigned char)ifr.ifr_addr.sa_data[i]);
            result = (result + buffer + ":");
        }
        cout<<"MAC address of this computer is: "<<result<<endl;
        return 0;
    }

    return 1;
}
```

SCREENSHOT:


Activities
Sublime Text 2
Thu Jul 25, 3:56 AM
~/CSN361/q2.cpp - Sublime Text 2 (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
q2.cpp
Line 4, Column 19
Spaces: 4
C++

```cpp
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <linux/if.h>
#include <netdb.h>
#include <iostream>
#include <string.h>
#include <bits/stdc++.h>

using namespace std;

int main()
{
    struct ifreq ifr;
    int fd = socket(PF_INET, SOCK_DGRAM, IPPROTO_IP);
    string ans;
    char buff[3];
    strcpy(ifr.ifr_name, "enp3s0");

    if (ioctl(fd, SIOCGIFHWADDR, &ifr) == 0) {
        for (int i = 0; i <= 5; i++){
            snprintf(buff, sizeof(buff), "%.2x", (unsigned char)ifr.ifr_addr.sa_data[i]);
            ans = ans + buff + ":";
        }
        cout << "MAC address: " << ans << endl;
        return 0;
    }

    return 1;
}
```

```
MAC address: d4:81:d7:8e:f7:0a:
[Finished in 1.1s]
```

QUESTION 3: Write your own version of ping program in C language.

SOLUTION:

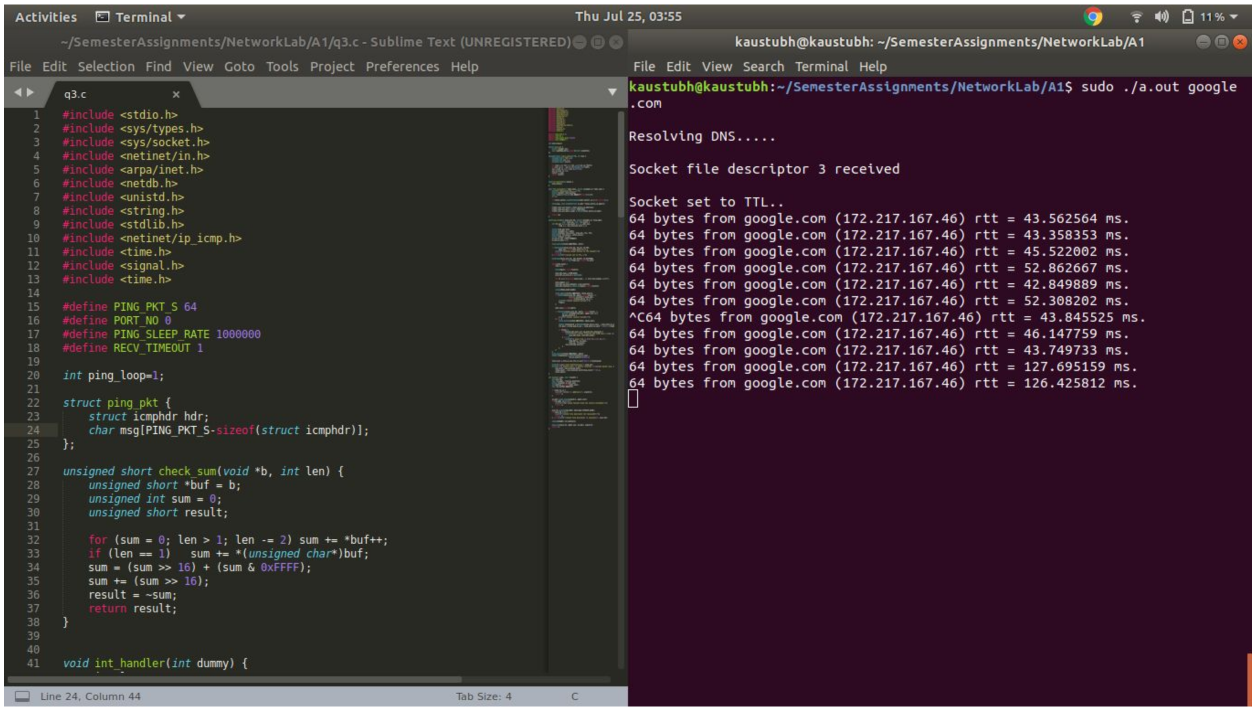
```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <netinet/ip_icmp.h>
#include <time.h>
#include <signal.h>
#include <time.h>

#define PING_PKT_S 64
#define PORT_NO 0
#define PING_SLEEP_RATE 1000000
#define RECV_TIMEOUT 1

int ping_loop=1;

struct ping_pkt {
      struct icmphdr hdr;
      char msg[PING_PKT_S-sizeof(struct icmphdr)];
};

unsigned short check_sum(void *b, int len) {
    unsigned short *buf = b;
      unsigned int sum = 0;
      unsigned short result;

      for (sum = 0; len > 1; len -= 2) sum += *buf++;
      if (len == 1)    sum += *(unsigned char*)buf;
      sum = (sum >> 16) + (sum & 0xFFFF);
      sum += (sum >> 16);
      result = ~sum;
```

```
	return result;
}


void int_handler(int dummy) {
	ping_loop=0;
}

char *dns_lookup(char *addr_host, struct sockaddr_in *addr_con) {
	printf("\nResolving DNS.....\n");
	struct hostent *host_entity;
	char *ip=(char*)malloc(NI_MAXHOST*sizeof(char));
	int i;

	if((host_entity=gethostbyname(addr_host))==NULL) return NULL;

	strcpy(ip,inet_ntoa(*(struct in_addr *)host_entity->h_addr));

	(*addr_con).sin_family = host_entity->h_addrtype;
	(*addr_con).sin_port = htons (PORT_NO);
	(*addr_con).sin_addr.s_addr = *(long*)host_entity->h_addr;

	return ip;
}

void ping_site(int ping_sock_fd, struct sockaddr_in *ping_addr,
                      char *ping_ip, char *rev_host) {
	int ttl_val = 64, msg_count = 0, i, addr_len,
	      flag = 1, msg_received_count = 0;

	struct ping_pkt pckt;
	struct sockaddr_in r_addr;
	struct timespec time_start, time_end, tfs, tfe;
	long double rtt_msec=0, total_msec=0;
	struct timeval tv_out;
	tv_out.tv_sec = RECV_TIMEOUT;
	tv_out.tv_usec = 0;

	clock_gettime(CLOCK_MONOTONIC, &tfs);
```

```
if (setsockopt(ping_sock_fd, SOL_IP, IP_TTL,
           &ttl_val, sizeof(ttl_val)) != 0) {
     printf("\nSetting socket options to TTL failed!\n");
     return;
} else printf("\nSocket set to TTL..\n");

setsockopt(ping_sock_fd, SOL_SOCKET, SO_RCVTIMEO,
                (const char*)&tv_out, sizeof tv_out);

while(ping_loop) {
     flag = 1;

     bzero(&pckt, sizeof(pckt));

     pckt.hdr.type = ICMP_ECHO;
     pckt.hdr.un.echo.id = getpid();

     for(i=0;i<sizeof(pckt.msg)-1;i++) pckt.msg[i]=i+'0';

     pckt.msg[i] = 0;
     pckt.hdr.un.echo.sequence = msg_count++;
     pckt.hdr.checksum = check_sum(&pckt, sizeof(pckt));

     usleep(PING_SLEEP_RATE);

     clock_gettime(CLOCK_MONOTONIC, &time_start);
     if (sendto(ping_sock_fd, &pckt, sizeof(pckt), 0,
                    (struct sockaddr*) ping_addr,
                     sizeof(*ping_addr)) <= 0) {
          printf("\nPacket Sending Failed!\n");
          flag=0;
     }

     addr_len=sizeof(r_addr);

     if (recvfrom(ping_sock_fd, &pckt, sizeof(pckt), 0,
              (struct sockaddr*)&r_addr, &addr_len) <= 0
              && msg_count>1) {
                printf("\nPacket receive failed!\n");
            } else {
                clock_gettime(CLOCK_MONOTONIC, &time_end);

                double timeElapsed
                =((double)(time_end.tv_nsec-time_start.tv_nsec))/1000000.0;

                rtt_msec
                =(time_end.tv_sec-time_start.tv_sec)*1000.0+timeElapsed;

                if (flag) {
                    if (!(pckt.hdr.type ==69 && pckt.hdr.code==0)) {
                        printf("Error..Packet received with ICMP type %d
                         code %d\n", pckt.hdr.type, pckt.hdr.code);
                    } else {
                        printf("%d bytes from %s (%s) rtt = %Lf ms.\n",
                            PING_PKT_S, rev_host,
                            ping_ip, rtt_msec);
                        msg_received_count++;
                    }
                }
            }
        }
        clock_gettime(CLOCK_MONOTONIC, &tfe);
        double timeElapsed = ((double)(tfe.tv_nsec -
                            tfs.tv_nsec))/1000000.0;

        total_msec = (tfe.tv_sec-tfs.tv_sec)*1000.0 + timeElapsed;

        printf("\n===%s ping statistics===\n", ping_ip);
        printf("\n%d packets sent,
            %d packets received,
            %f percent packet loss.
            Total time: %Lf ms.\n\n",
            msg_count, msg_received_count,
            ((msg_count - msg_received_count)/msg_count)*100.0,
            total_msec);
}
```

```
int main(int argc, char *argv[]) {
	int sock_fd;
	char *ip_addr, *reverse_hostname;
	struct sockaddr_in addr_con;
	int addrlen = sizeof(addr_con);
	char net_buf[NI_MAXHOST];

	if (argc != 2) {
		printf("\nFormat %s <address>\n", argv[0]);
		return 0;
	}

	ip_addr = dns_lookup(argv[1], &addr_con);
	if (ip_addr == NULL) {
		printf("\nDNS lookup failed! Could not resolve hostname!\n");
		return 0;
	}

	sock_fd = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP);
	if (sock_fd < 0) {
		printf("\nSocket file descriptor not received!!\n");
		return 0;
	} else printf("\nSocket file descriptor %d received\n", sock_fd);

	signal(SIGINT, int_handler);

	ping_site(sock_fd, &addr_con, ip_addr, argv[1]);
	return 0;
}
```

SCREENSHOT:


Activities
Terminal
Thu Jul 25, 03:55
11 %
~/SemesterAssignments/NetworkLab/A1/q3.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
q3.c
Line 24, Column 44
Tab Size: 4
C
kaustubh@kaustubh: ~/SemesterAssignments/NetworkLab/A1
File Edit View Search Terminal Help

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <netinet/ip_icmp.h>
#include <time.h>
#include <signal.h>
#include <time.h>

#define PING_PKT_S 64
#define PORT_NO 0
#define PING_SLEEP_RATE 1000000
#define RECV_TIMEOUT 1

int ping_loop=1;

struct ping_pkt {
    struct icmphdr hdr;
    char msg[PING_PKT_S-sizeof(struct icmphdr)];
};

unsigned short check_sum(void *b, int len) {
    unsigned short *buf = b;
    unsigned int sum = 0;
    unsigned short result;

    for (sum = 0; len > 1; len -= 2) sum += *buf++;
    if (len == 1)   sum += *(unsigned char*)buf;
    sum = (sum >> 16) + (sum & 0xFFFF);
    sum += (sum >> 16);
    result = ~sum;
    return result;
}


void int_handler(int dummy) {
```

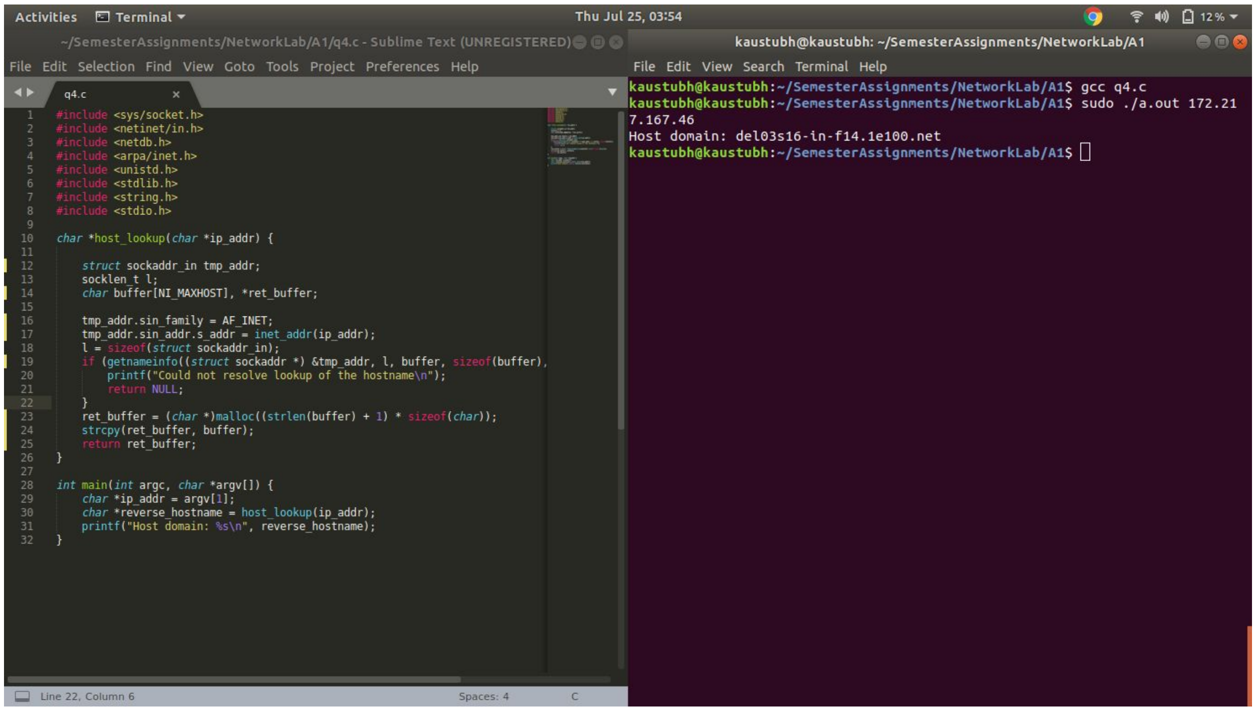
```
kaustubh@kaustubh:~/SemesterAssignments/NetworkLab/A1$ sudo ./a.out google
.com

Resolving DNS.....

Socket file descriptor 3 received

Socket set to TTL..
64 bytes from google.com (172.217.167.46) rtt = 43.562564 ms.
64 bytes from google.com (172.217.167.46) rtt = 43.358353 ms.
64 bytes from google.com (172.217.167.46) rtt = 45.522002 ms.
64 bytes from google.com (172.217.167.46) rtt = 52.862667 ms.
64 bytes from google.com (172.217.167.46) rtt = 42.849889 ms.
64 bytes from google.com (172.217.167.46) rtt = 52.308202 ms.
^C64 bytes from google.com (172.217.167.46) rtt = 43.845525 ms.
64 bytes from google.com (172.217.167.46) rtt = 46.147759 ms.
64 bytes from google.com (172.217.167.46) rtt = 43.749733 ms.
64 bytes from google.com (172.217.167.46) rtt = 127.695159 ms.
64 bytes from google.com (172.217.167.46) rtt = 126.425812 ms.
```

QUESTION 4: Write C program to find host name from IP address.

SOLUTION:

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

char *host_lookup(char *ip_addr) {

    struct sockaddr_in tmp_addr;
    socklen_t l;
    char buffer[NI_MAXHOST], *ret_buffer;

    tmp_addr.sin_family = AF_INET;
    tmp_addr.sin_addr.s_addr = inet_addr(ip_addr);
    l = sizeof(struct sockaddr_in);
    if (getnameinfo((struct sockaddr *)
        &tmp_addr, l, buffer, sizeof(buffer),
         NULL, 0, NI_NAMEREQD)) {
        printf("Could not resolve lookup of the hostname\n");
        return NULL;
    }
    ret_buffer=(char *)malloc((strlen(buffer) + 1)*sizeof(char));
    strcpy(ret_buffer, buffer);
    return ret_buffer;
}

int main(int argc, char *argv[]) {
    char *ip_addr = argv[1];
    char *reverse_hostname = host_lookup(ip_addr);
    printf("Host domain: %s\n", reverse_hostname);
}
```

SCREENSHOT:



```c
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

char *host_lookup(char *ip_addr) {

    struct sockaddr_in tmp_addr;
    socklen_t l;
    char buffer[NI_MAXHOST], *ret_buffer;

    tmp_addr.sin_family = AF_INET;
    tmp_addr.sin_addr.s_addr = inet_addr(ip_addr);
    l = sizeof(struct sockaddr_in);
    if (getnameinfo((struct sockaddr *) &tmp_addr, l, buffer, sizeof(buffer),
        printf("Could not resolve lookup of the hostname\n");
        return NULL;
    }
    ret_buffer = (char *)malloc((strlen(buffer) + 1) * sizeof(char));
    strcpy(ret_buffer, buffer);
    return ret_buffer;
}

int main(int argc, char *argv[]) {
    char *ip_addr = argv[1];
    char *reverse_hostname = host_lookup(ip_addr);
    printf("Host domain: %s\n", reverse_hostname);
}
```

```
kaustubh@kaustubh:~/SemesterAssignments/NetworkLab/A1$ gcc q4.c
kaustubh@kaustubh:~/SemesterAssignments/NetworkLab/A1$ sudo ./a.out 172.21
7.167.46
Host domain: del03s16-in-f14.1e100.net
kaustubh@kaustubh:~/SemesterAssignments/NetworkLab/A1$
```